Dynamic Model Formulation and Calibration for Wheeled Mobile Robots

Neal A. Seegmiller

CMU-RI-TR-14-27

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Alonzo Kelly (Chair)
David Wettergreen
George Kantor
Karl Iagnemma

Defended on: October 23, 2014

Keywords: wheeled mobile robots, kinematics, dynamics, wheel slip, wheel-terrain interaction, calibration, model identification, motion planning, model predictive control

dedicated to my parents, who supported my academic pursuits from the beginning

Abstract

Advances in hardware design have made wheeled mobile robots (WMRs) exceptionally mobile. To fully exploit this mobility, WMR planning, control, and estimation systems require motion models that are fast and accurate. Much of the published theory on WMR modeling is limited to 2D or kinematics, but 3D dynamic (or force-driven) models are required when traversing challenging terrain, executing aggressive maneuvers, and manipulating heavy payloads. This thesis advances the state of the art in both the formulation and calibration of WMR models

We present novel WMR model formulations that are high-fidelity, general, modular, and fast. We provide a general method to derive 3D velocity kinematics for any WMR joint configuration. Using this method, we obtain constraints on wheel-ground contact point velocities for our differential algebraic equation (DAE)-based models. Our "stabilized DAE" kinematics formulation enables constrained, drift-free motion prediction on rough terrain. We also enhance the kinematics to predict nonzero wheel slip in a principled way based on gravitational, inertial, and dissipative forces. Unlike ordinary differential equation (ODE)-based dynamic models which can be very stiff, our constrained dynamics formulation permits large integration steps without compromising stability. Some alternatives like Open Dynamics Engine also use constraints, but can only approximate Coulomb friction at contacts. In contrast, we can enforce realistic, nonlinear models of wheel-terrain interaction (e.g. empirical models for pneumatic tires, terramechanics-based models) using a novel force-balance optimization technique.

Simulation tests show our kinematic and dynamic models to be more functional, stable, and efficient than common alternatives. Simulations run 1K-10K$\times$ faster than real-time on an ordinary PC, even while predicting articulated motion on rough terrain and enforcing realistic wheel-terrain interaction models.

In addition, we present a novel Integrated Prediction Error Minimization (IPEM) method to calibrate model parameters that is general, convenient, online, and evaluative. Ordinarily system dynamics are calibrated by minimizing the error of instantaneous output predictions. IPEM instead forms predictions by integrating the system dynamics over an interval; benefits include reduced sensing requirements, better observability, and accuracy over a longer horizon. In addition to calibrating out systematic errors, we simultaneously calibrate a model of stochastic error propagation to quantify the uncertainty of motion predictions.

Experimental results on multiple platforms and terrain types show that parameter estimates converge quickly during online calibration, and uncertainty is well-characterized. Under normal conditions, our enhanced kinematic model can predict nonzero wheel slip as accurately as a full dynamic model for a fraction of the computation cost. Finally, odometry is greatly improved when using IPEM vs. manual calibration, and when using 3D vs. 2D kinematics. To facilitate their use, we have released open source MATLAB and C++ libraries implementing the model formulation and calibration methods in this thesis.

Acknowledgments

Many people at Carnegie Mellon University supported me in my thesis research. David Wettergreen gave me interesting problems to consider as a Masters student, which led me towards my thesis topic. Al Kelly gave me lots of early guidance, based on his extensive experience getting WMRs to work in the real world, then freedom to define and explore my topic. Forrest Rogers-Marcovitz's prior work on vehicle model identification provided a foundation for my calibration research. Forrest, Venkat Rajagopalan, David Kohanbash, and members of the UPI project (UGCV PerceptOR Integrated) contributed by collecting experimental data used to validate my methods.

This research was made with U.S. government support under and awarded by two fellowships: the DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate Fellowship (32 CFR 168a) and the National Science Foundation Graduate Research Fellowship (0946825). Additional project support was provided by the Army Research Office (W911NF-09-1-0557) and the Army Research Laboratory (W911NF-10-2-0016).

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Advances in hardware design have made wheeled mobile robots (WMRs) exceptionally mobile. For example, the rocker-bogie design of NASA's Spirit and Opportunity rovers enables them to traverse obstacles larger than their wheel diameter. Furthermore, their lightweight wheels use a spiral flecture to absorb shock and cleats for traction in soft soil [78]. The advanced suspension on NREC's Autonomous Platform Demonstrator (APD) enables high-speed maneuverability and the navigation of 60-percent slopes [93]. Often, however, WMRs do not fully exploit their mobility when operating autonomously, but instead drive conservatively. One reason is that their planning, control, and estimation systems require motion models that are both *fast* and *accurate*, but these are difficult to produce.

1.1 Problem Description

Motion models predict changes in state given a sequence of inputs and information about the WMR and its environment. The winning Tartan Racing team in the DARPA Urban Challenge observed that model fidelity is strongly correlated to the effectiveness of model predictive planning (MPP) [130]. For autonomous WMRs to plan trajectories that fully exploit their mobility, they require high-fidelity, 3D dynamic models. To plan such trajectories in real-time, the models must also be computationally cheap. There are two primary challenges to producing such models: formulation and calibration.

1.1.1 Challenge 1: Model Formulation

By model formulation we refer to expression of the model as a system of mathematical equations, and the specification of algorithms to solve those equations and predict motion. Roboticists have modeled WMRs for decades and their models vary widely in complexity (Section 2.1). Their models can be categorized as planar or 3D, and kinematic or dynamic.

Generally speaking, a "dynamic model" (as referred to by the title of this thesis) is a mathematical model that describes the behavior of a system over time; but, hereafter we will typically use "dynamic model" more restrictively to denote a second-order, force-driven model that accounts for inertia. We will use "kinematic model" to denote a first-order, velocity-driven model.

Most fielded WMRs today make do with basic 2D kinematic models for the following reasons:

First, they are easy to derive. Planar models for differential drive robots and Ackerman-steered cars are already available in the literature. Analytic 3D dynamic models for new WMRs are hard to derive from scratch. One could use a general purpose physics simulator, but it may have limited functionality for wheel-ground contact modeling.

Second, they are computationally cheap. Real-time MPP requires simulation *orders of magnitude* faster than real-time on a fraction of a processor, because many candidate trajectories must be evaluated to choose the best one for execution. There is a prevalent tradeoff between model fidelity and speed in related work, causing some to consider 3D dynamics to be too computationally expensive for practical use [47].

Finally, 2D kinematic models are sometimes sufficient when WMRs are driven conservatively, but challenging scenarios require high-fidelity models. For example, when traversing very steep or rough terrain WMRs must reason about suspension deflections, wheel-terrain interaction (i.e. slip, traction limits), and the risk of rollover. During aggressive maneuvers WMRs must additionally account for powertrain/actuator characteristics (e.g. maximum power, torque) and ballistic motion. Manipulating heavy payloads (towing, plowing, lifting, etc.) exerts forces on WMRs and changes their mass distribution, affecting mobility and stability.

To address these needs, we require a WMR model formulation that is:

- high-fidelity: accounts for 3D articulations, wheel-terrain interaction, actuator characteristics, and liftoff
- general: supports any WMR joint configuration through systematic methods
- modular: allows different wheel-ground contact and actuator models to be swapped in
- fast: can be simulated orders of magnitude faster than real-time

1.1.2 Challenge 2: Model Calibration

A detailed model alone does not guarantee accuracy; to accurately predict the motion of a real system the model's parameters must be correctly identified. These parameters include dimensions, mass properties, and constants in the wheel-ground contact and actuator models.

Common approaches to WMR model identification include: using nominal values, taking manual measurements, analyzing subsystems in isolation, or performing calibration routines which may require special trajectories (Section 3.1). These procedures can be laborious and inaccurate. Subsystems may be well-calibrated while the system as a whole is not. Motion may be predicted well for the trajectories and terrain types encountered during calibration, but not others encountered in the field.

Some have proposed convenient odometry calibration methods, but these are mostly limited to planar, differential drive robots; they are not applicable to the high-fidelity models specified in Section 1.1.1. Furthermore, most methods do not quantify the uncertainty of model predictions, even though modern probabilistic planners and estimators require this information.

To address these shortcomings, we require a WMR model calibration method that is:

- general: applicable to any high-fidelity WMR model
- convenient: does not require special trajectories or continuous sensing

- online: adapts automatically to changing conditions
- evaluative: quantifies the uncertainty of motion predictions

1.2 Overview of Solution

In the proposal for this thesis, we hypothesized that a WMR model formulation could meet all the requirements in Section 1.1.1. We successfully developed such a formulation and present it in Chapter 2. We present novel differential algebraic equation (DAE)-based formulations of both WMR kinematics and dynamics. Our "stabilized DAE" kinematics formulation enables constrained, drift-free motion prediction on rough terrain (Section 2.2). We also enhance the kinematics to predict nonzero wheel slip (Section 2.3). Unlike ordinary differential equation (ODE)-based dynamic models which can be very stiff, our constrained dynamics formulation permits large integration steps without compromising stability (Section 2.4). Also, unlike Open Dynamics Engine which can only approximate Coulomb friction at contacts, we use a "force-balance optimization" technique to enforce realistic, nonlinear models of wheel-terrain interaction. Simulation tests demonstrate our formulations to be more functional, stable, and efficient than state-of-the-art alternatives (Section 2.5).

We also hypothesized that a calibration method could meet all the requirements in Section 1.1.2. We developed such a method and present it in Chapter 3. Our solution is based on "Integrated Prediction Error Minimization" (or IPEM) which involves the integration of motion predictions over extended intervals. We both calibrate out systematic errors, and simultaneously calibrate a model of stochastic error propagation for uncertainty estimation. We assess the accuracy of our calibrated, high-fidelity models in physical experiments on a variety of platforms and terrain types (Section 3.4); these results validate both our model formulations and calibration method.

Related work on WMR model formulation and calibration is covered their respective chapters. To avoid overuse of the passive voice, this document uses "we" and "our" when explaining this research. However, care is taken to distinguish the author's individual contributions from those made in collaboration with others.

At the end of this document, a glossary is provided for frequently-used technical terms. Details necessary for the implementation of this research, but not critical to the theoretical contribution, are provided in the appendices.

1.3 Notational Conventions

This section briefly explains the notational conventions used in mathematical expressions in this document. Table 1.1 defines commonly used symbols, and Table 1.2 defines commonly used variable names.

Three types of vectors are distinguished in this document: general vectors of any size, three-element Cartesian vectors, and six-element Plücker coordinate or spatial vectors. Spatial vectors can represent either motion or force, and have both angular and linear components. They are

used in spatial vector algebra (SVA) for the concise expression of rigid body dynamics equations [36]. A more thorough explanation of spatial vector algebra is provided in Appendix B.

The skew-symmetric matrix notation explicitly means:

$$[\vec{u}]_\times = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \tag{1.1}$$

Skew-symmetric matrices are used to compute cross products via matrix multiplication: $\vec{u} \times \vec{v} = [\vec{u}]_\times \vec{v}$.

The expression $f()$ means "a function of" the inputs in parentheses. The name f is often used generically for unspecified functions. One exception is the repeated use of f to denote a specific function in the system dynamics (3.1) throughout much of Chapter 3. The names of *vector-valued* functions are formatted with vector notation. For example, a function named $g()$ has a scalar range whereas $\mathbf{g}()$ has a vector range.

Subscripts and superscripts are used to indicate frames of reference for physical quantities. For example, $\vec{v}_a^b$ denotes the linear velocity of frame a relative to frame b. While this vector has physical meaning independent of any coordinate system, a coordinate system must be chosen to express it numerically. ${}^c\vec{v}_a^b$ indicates that the velocity is expressed in the coordinates associated with frame c. The phrase "local coordinates" is used as follows: $\vec{v}_a^b$ in local coordinates implies ${}^a\vec{v}_a^b$ (the velocity of frame a expressed in the coordinates of frame a). In related work this is sometimes called "link coordinates" [37]. Sometimes the coordinate system is left unspecified or to be inferred from context; in this case, note that quantities must be expressed in the same coordinates to be combined in numerical operations.

Scripts are also used for rotation matrices, homogeneous transforms, and Plücker transforms to denote the change in coordinates effected by left multiplication. For example, $\vec{u}$ is transformed from a to b coordinates as follows: ${}^b\vec{u} = R_a^b({}^a\vec{u})$. As noted in Table 1.1 and further explained in Appendix B, the effect of left multiplication by Plücker transforms is different for spatial motion and force vectors.

Array indexing in this document begins at 1 not 0. Sets of indices will typically be named with a descriptive word, formatted with a smaller font size (e.g. wheel for the indices of wheel constraints). Standard set operator symbols are used.

Pose vectors ($\boldsymbol{\rho}$) and homogeneous/Plücker transforms (T/X) encode the same information: orientation and position. Orientation is represented by either Euler angles or a quaternion in a pose vector, and by a rotation matrix in a homogeneous/Plücker transform. See Appendix A for details on converting between orientation representations.

The WMR state vector $\mathbf{q}$ and joint space velocity/acceleration vectors ($\dot{\mathbf{q}}'/\ddot{\mathbf{q}}'$) are explained in Section 2.2.1.

Table 1.1: Common mathematical symbols

Category	Symbol	Meaning
Vector/Matrix[1]	$\mathbf{u}$	(bold lowercase) column vector of any size.
	$\vec{u}$	(overset harpoon) 3-element Cartesian coordinate vector
	$\vec{u}$	(overset arrow) 6-element Plücker coordinate/spatial vector
	$[\mathbf{u}; \mathbf{u}]$	(semicolon) concatenate vertically into a column vector
	$\|\mathbf{u}\|$	Euclidean norm, $\sqrt{\mathbf{u}^T\mathbf{u}}$
	M	(uppercase) matrix
	$[\vec{u}]_\times$	skew-symmetric matrix formed from a Cartesian coordinate vector, used for cross products
	$\mathrm{diag}(\mathbf{u})$	diagonal matrix formed from a vector
	M^+	pseudoinverse
Functions	$f()$	"a function of" the variables in parentheses
	$\mathbf{g}()$	(name is vector formatted) a vector-valued function
Scripts	${}^c u_a^b$	(scalar/vector) the quantity u is of frame a relative to frame b, expressed in the coordinates associated with frame c
	R_a^b	(rotation matrix) represents the orientation of frame a relative to frame b, transforms Cartesian coordinate vectors: ${}^b\vec{u} = R_a^b({}^a\vec{u})$
	T_a^b	(homogeneous transform) represents the position and orientation of frame a relative to frame b
Indexing	$\mathbf{u}(i)$	the i^{th} element of the vector $\mathbf{u}$, sometimes denoted u_i
	$\mathbf{u}(i...i+n)$	the subvector of elements i through $i+n$ of $\mathbf{u}$
	$M(i,j)$	the element of matrix A at row i, column j
	$M(i,*)$	the i^{th} row of matrix M
	$M(*,j)$	the j^{th} column of matrix M
	$M([i],[j])$	block (i,j) of the block matrix M
	$\mathbf{u}[k]$	the value of $\mathbf{u}$ at time step k, sometimes denoted $\mathbf{u}_k$
Set Operations	$x \in A$	x is an element of the set A
	$A \subseteq B$	A is a subset of B
	$A \cap B$	set intersection containing all elements in both A and B
	$A \cup B$	set union containing all elements in either A or B
	$A \setminus B$	set difference containing all elements of A not in B
	$A := \{x : ()\}$	(set builder notation) the set A contains all elements x for which the condition on x in $()$ is true
SVA[2]	X_a^b	(Plücker transform) transforms spatial *motion* vectors as follows: ${}^b\vec{m} = X_a^b({}^a\vec{m})$
	$(X_a^b)^{-T}$	(inverse transpose of Plücker transform) transforms spatial *force* vectors: ${}^b\vec{f} = (X_a^b)^{-T}({}^a\vec{f})$
	$\times_m$, $\times_f$	operators for cross products with motion and force vectors

[1] u and M are dummy variable names in this table.

[2] see Appendix B for details on Spatial Vector Algebra.

Table 1.2: Common variable names

Category	Variable	Meaning
Cartesian	$\vec{r}$	position/translation
(3×1)	$\vec{v}$	linear velocity
	$\vec{a}$	linear acceleration
	$\vec{\omega}$	angular velocity
	$\vec{\alpha}$	angular acceleration
	$\vec{f}$	linear force
	$\vec{\tau}$	torque/moment
Spatial	$\vec{v}$	spatial velocity $[\vec{\omega}; \vec{v}]$
(6×1)	$\vec{a}$	spatial acceleration $[\vec{\alpha}; \vec{a}]$
	$\vec{f}$	spatial force $[\vec{\tau}; \vec{f}]$
Transform	R	3×3 rotation matrix
	T	4×4 homogeneous transform matrix
	X	6×6 Plücker transform matrix
Zeros	$\mathbf{0}$	a vector of zeros
	$[0]$	a matrix or block of zeros
State	$\mathbf{o}$	orientation vector, Euler angles or quaternion
	$\boldsymbol{\rho}$	pose vector, comprises orientation and position $[\mathbf{o}; \vec{r}]$
	$\mathbf{q}$	state vector, comprises pose of body frame and joint displacements $[\boldsymbol{\rho}_b^w; \boldsymbol{\theta}]$
	$\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$	the first and second time derivatives of $\mathbf{q}$
	$\dot{\mathbf{q}}'$	joint space velocity, comprises the spatial velocity of the body frame and joint rates $[{}^b\vec{v}_b^w; \dot{\boldsymbol{\theta}}]$
	$\ddot{\mathbf{q}}'$	joint space acceleration, the time derivative of $\dot{\mathbf{q}}'$

Chapter 2

Model Formulation

This chapter presents our novel WMR model formulations. First, Section 2.1 discusses related work. Section 2.2 explains our constraint-based kinematics formulation, and Section 2.3 an enhancement to the kinematics to account for wheel slip. Section 2.4 explains our dynamics formulation, which handles wheel-ground contact and actuator forces via constraints. In Section 2.5 both formulations are evaluated in simulation, demonstrating improved functionality, stability, and computation speed over alternatives. Evaluation of predictive accuracy in physical experiments (after calibration) is presented in Section 3.4 of the following chapter.

2.1 Related Work on Model Formulation

This section presents related work on WMR kinematics (Section 2.1.1) and dynamics (Section 2.1.2). Section 2.1.3 categorizes this work and highlights novel features of our solution. First we define some relevant terminology.

Kinematics and dynamics terms. For manipulators, *forward* kinematics is the problem of calculating end effector pose given joint displacements. Displacements are angles for revolute joints and distances for prismatic joints. *Inverse* kinematics is the (usually more difficult) problem of calculating joint displacements given end effector pose. For WMRs, the *velocity kinematics*, which relate wheel (or joint) rates to body frame velocity, are of greater concern. The terms "forward" and "inverse" are ambiguous for WMR kinematics (either the wheels or body frame can be considered to be the end effector). Usually one must solve hybrid kinematics with some known and unknown quantities. For example, Tarokh and McDermott [125] define "navigation" and "actuation" kinematics. In navigation kinematics (relevant to simulation and estimation) one solves for the body velocity and passive joint rates given actuated joint rates. In actuation kinematics (relevant to control) one solves for actuated joint rates given a desired body velocity.

Forward *dynamics* refers to the calculation of system motion given input forces, and inverse dynamics refers to the opposite. For WMRs, forward dynamic (and velocity kinematic) calculations typically require the resolution of overconstraint issues. For example, a skid-steer vehicle cannot turn without wheel slip.

Much of the literature on WMR kinematics and dynamics provides only *ad hoc* derivations

for simple platforms, such as planar differential drive robots. Some provide *general*, systematic methods to derive the velocity kinematics of any platform. These derivations can be categorized by their mathematical basis:

- geometry-based: equations relating distances between points and angles between line segments are generated by inspection. Planar derivations may use the instantaneous center of rotation (ICR). Geometry-based derivations are usually not general.
- transform-based: Chains of homogeneous transforms are constructed for each wheel, then differentiated to obtain velocity kinematics. A general approach.
- velocity propagation-based: Velocity kinematics are computed directly, without differentiation, by propagating velocities frame to frame. Includes twist and screw methods. Some use the adjoint operator to transform velocity between frames once, but not recursively for arbitrary kinematic chains.

Simple 2D models can often be derived *analytically* or *symbolically*; however, analytic derivations can be cumbersome for 3D models of complex WMRs. Such models are simpler and faster to simulate *numerically*.

Constraint terms. Unlike robotic manipulators, WMR motion is governed by both *holonomic* and *nonholonomic* constraints. Holonomic constraints are of the form:

$$c(\mathbf{x}, t) = 0 \tag{2.1}$$

They are only a function of state ($\mathbf{x}$) and possibly time. For WMRs, the constraint that wheels not penetrate the terrain surface is holonomic. In contrast, nonholonomic constraints are of the form:

$$c(\dot{\mathbf{x}}, \mathbf{x}, t) = 0 \tag{2.2}$$

They are velocity-dependent. For WMRs, the constraint that wheels roll without slipping or not slip laterally are nonholonomic. Mobile robots with nonholonomic constraints are underactuated, because not all degrees of freedom (DOF) of body frame motion are controllable. For example, a car cannot move sideways to parallel park, but must instead execute a forward/backward translation and steering maneuver.

Differential equation terms. System dynamics are typically expressed as *ordinary differential equations* (ODEs). An ODE is an equation containing a function of one independent variable (e.g. time) and its derivatives. In this document the function will often be vector-valued; but, we will still refer to the equation as an "ODE" rather than a "system of ODEs" like in some related work. A first-order, explicit ODE is of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \tag{2.3}$$

A second-order, explicit ODE is of the form:

$$\ddot{\mathbf{x}}(t) = \mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), t) \tag{2.4}$$

Internally, $\mathbf{f}()$ may also be a function of inputs $\mathbf{u}(t)$ and parameters $\mathbf{p}$ (Section 3.1.1).

Alternatively, WMR system dynamics can be expressed as *differential algebraic equations* (DAEs). Generally, DAEs are of the form:

$$\mathbf{F}(\dot{\mathbf{x}}(t), \mathbf{x}(t), t) = \mathbf{0} \tag{2.5}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{F} : \mathbb{R}^{2n+1} \to \mathbb{R}^n$. Unlike for an ODE, the equation is not algebraically solvable for all components of $\dot{\mathbf{x}}$. A semi-explicit DAE is of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{y}(t), t) \tag{2.6}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}(t), \mathbf{y}(t), t) \tag{2.7}$$

where:

$$\mathbf{x}(t) \in \mathbb{R}^n \qquad \mathbf{f} : \mathbb{R}^{n+m+1} \to \mathbb{R}^n$$

$$\mathbf{y}(t) \in \mathbb{R}^m \qquad \mathbf{g} : \mathbb{R}^{n+m+1} \to \mathbb{R}^m$$

The semi-explicit DAE comprises two parts: a differential equation (2.6) and an algebraic constraint equation (2.7). Every solution of the constraint equation defines a unique direction for $\dot{\mathbf{x}}$ via the differential equation, but not every point $(\mathbf{x}(t), \mathbf{y}(t), t)$ is a solution of the constraint equation. Solving a DAE is more complex than solving an ODE because initial conditions and evolution of the state must be consistent with constraints. DAE models of WMRs include both holonomic and nonholonomic constraints. Differentiation of the holonomic constraints with respect to time (index reduction) can be performed to convert DAEs to implicit ODEs.

Both our kinematics and dynamics formulations use DAEs to predict motion. Note that the *kinematics* are drift-free, meaning that if control inputs are zero the WMR does not move. Furthermore, for most WMRs there are sufficient constraints to uniquely define motion. Accordingly, kinematic motion prediction merely requires solving a constraint equation, i.e. (2.33) in Section 2.2.5. In contrast, the *dynamics* are not drift-free; inertia and gravity cause the WMR to move even when control inputs are zero. Accordingly, the constrained dynamics equation (2.65) has clearly distinguished differential and algebraic parts (Section 2.4.2).

2.1.1 Related Work on WMR Kinematics

2D kinematics. Early work on mobile robot kinematics dealt only with 2D/planar models. Muir and Neuman [89] provided the first general method to derive WMR kinematics. They use a similar approach to that used for manipulators, while noting that WMRs are distinguished from manipulators by closed-link chains and "higher-pair" wheel-ground contacts. They assign frames in a transformation graph following the Sheth-Uicker convention, cascade transforms relating wheel to body positions, then symbolically differentiate them to obtain "wheel Jacobians" relating wheel to body velocities.

Several authors made modifications to Muir and Neuman's method. Whereas Muir and Neuman used the pseudoinverse to resolve overconstraint due to wheel misalignment, Alexander and Maddocks [4] instead used a convex dissipation functional based on Coulomb's law of friction. Rajagopalan [99] extended the method to handle an inclined steering column. Shin and Park [113] revised the method; representing wheel rotation and wheel-ground contact more realistically with individual lower pairs instead of a single planar pair.

Others provided geometry-based, ad hoc kinematics derivations for special configurations. Agullo et al. [2] derived the kinematics for "directional sliding" wheels (also called "omni," "Swedish," or "Mecanum" wheels). Vehicles equipped with such wheels are capable of 3-DOF planar motion, meaning that yaw and translation can be controlled independently. Burke and Durrant-Whyte [20] derived the kinematics of modular or reconfigurable WMRs. Borenstein

[16] derived the kinematics of a WMR with a compliant linkage, designed to mitigate wheel misalignment due to motor speed limits. Kim et al. [67][68] derived the kinematics for an omnidirectional WMR with three steerable wheels and proposed a basic path generation technique.

Some categorized WMR types by mobility. Kanayama [59] defined types based on the steerability of two wheels: FF fixed-fixed (e.g. differential drive), FS fixed-steered (e.g. a bicycle), and SS steered-steered (omnidirectional). Campion et al. [21] defined five types based on a "degree of mobility" and "degree of steerability."

More recently, some have proposed alternative *general* approaches to 2D kinematics derivation. Kim et al. [69] proposed an approach based on the transfer method for parallel manipulators, in which pseudo-joints are added at the wheel ground contacts. Fu and Krovi [38] extend a twist-based approach, used previously for parallel manipulators and multifinger grasping, to derive the kinematics of a reconfigurable WMR.

3D kinematics. In 2005 Tarokh and McDermott [125] provided the first general method to derive 3D WMR kinematics, which account for suspension articulations and uneven terrain. Their method is similar to Muir and Neuman's; they assign frames following the Denavit-Hartenberg (D-H) convention, construct homogeneous transform chains from body to wheel-ground contact frames, then symbolically differentiate them to obtain wheel Jacobians. They demonstrated their method for the Rocky 7 rocker-bogie rover (Figure 2.2).

In 2006 Chang et al. [24] noted that terms could be rearranged to simplify Tarokh and McDermott's derivation of the Rocky 7 kinematics. Skew-symmetric matrices appear in the simplified derivation, and the need for symbolic differentiation is eliminated. Later, Chang et al. [25] formalized this work into a "wheel-center modeling" method. In 2007, Tarokh and Mcdermott [126] also proposed a velocity propagation-based alternative to their prior transform-based approach. Provided with just a D-H parameter table, velocity kinematics could be computed numerically or symbolically (using a symbolic software tool). Recently, Tarokh et al. [127] utilized their kinematics approach for balance control of a complex rover with 3-DOF legs for each wheel.

Both Chang et al. and Tarokh et al.'s methods propagate velocities using a relation that includes the cross product of angular velocity and position. Kelly [62] explains why by presenting a vector algebra formulation of WMR velocity kinematics based on the "Transport Theorem." This theorem is also the basis for recursive Newton-Euler equations previously developed for manipulators [81]. By using the Transport Theorem, Kelly's formulation is coordinate-free and extensible to acceleration and higher-order kinematics [65]. We utilized this formulation to derive the 3D kinematics of the Zoë rover in [64].

Other authors have modeled 3D WMR kinematics for a particular design or application without providing general methods. Balaram [8] derived the 3D kinematics of the Rocky 7 rover before Tarokh and McDermott, but didn't explain how, or handle navigation kinematics (i.e. solution for the body frame velocity) except indirectly through a Kalman filter. Le Menn et al. [77] and Benamar et al. [12] present a reciprocal screw-based approach for multi-monocycle robots. Choi and Sreenivasan [26] analyze the use of a variable length axle to enable rolling without slipping on uneven terrain. Chakraborty and Ghosal [23] instead propose using a passive variable camber joint and torus-shaped wheels to avoid slip. Lamon and Siegwart [75] provide rare experimental results on 3D odometry for their SOLERO platform, but their geometry-based kinematics derivation is not general.

Constrained simulation. While Tarokh and McDermott [125] provided a ground-breaking

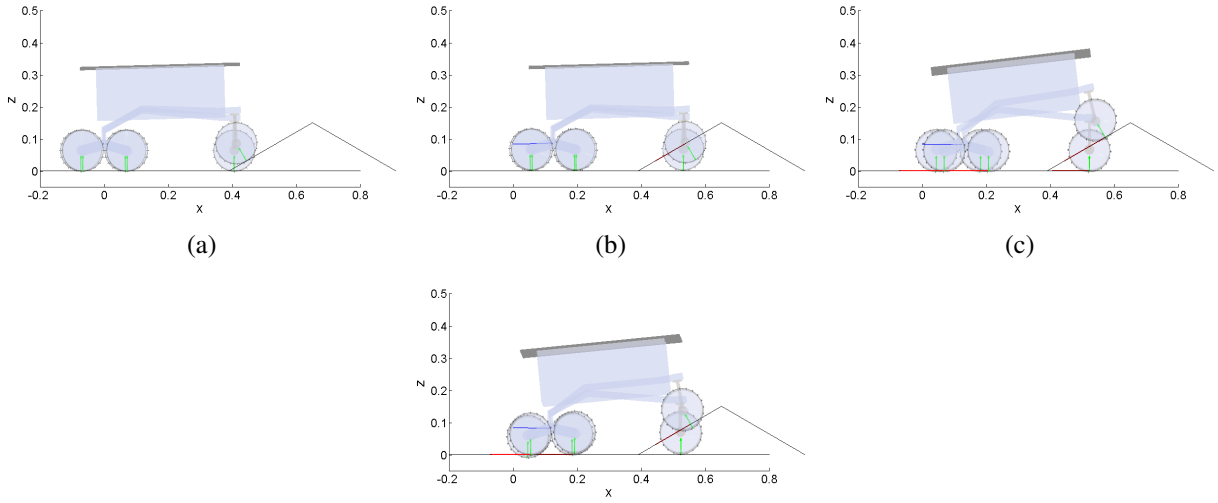


Figure 2.1: Figures (a) through (c) depict *unconstrained* kinematic motion prediction: (a) The rover state at time i. (b) The rover state at time $i + 1$ after taking an unconstrained step (i.e. based on nominal velocity or 2D kinematics). The front right wheel penetrates the ramp. (c) The rover state at time $i + 1$ after "terrain fitting," which induces unwanted wheel slip. Figure (d) depicts the predicted rover state at time $i + 1$ using our *constrained* method. No terrain fitting is required.

method to derive WMR kinematic models, their method to simulate these models has disadvantages. At each time step they execute an unconstrained motion, then solve for elevation, attitude, and passive joint displacements by minimization of a nonlinear "contact error" objective function. We will refer to this minimization procedure as "terrain fitting." In 2013, Tarokh et al. presumably still used this procedure to make "adjustments for the rover to conform to the terrain" [127]. Howard and Kelly [46] also used terrain fitting in motion models for their rough terrain trajectory generation algorithm. Terrain fitting keeps wheels in contact with the terrain, but it is computationally expensive, and it affects translation tangent to the terrain surface such that wheel slip cannot be effectively controlled.

A more elegant and efficient solution is to directly solve for vehicle motions that respect holonomic and nonholonomic constraints in one step. Saha and Angeles [105] used the orthogonal complement to the nonholonomic constraint matrix to derive constraint-free 2D kinematics. Choi and Sreenivasan [26] and Chakraborty and Ghosal [23] derived constraint-free kinematics in 3D using a differential algebraic equation (DAE) formulation. They differentiated holonomic constraints with respect to time to convert the DAE to an ordinary differential equation (ODE). Eathakota et al. [34] and Gattupalli et al. [39] likewise used DAEs, but only for their particular platform. Our work is the first to combine a general method to derive WMR kinematics and a DAE formulation for motion prediction. The difference between constrained vs. unconstrained kinematic motion prediction is illustrated in Figure 2.1 and further discussed in Sections 2.2.4 and 2.2.5.

Shortcomings. None of these kinematic models accounts for wheel slip, but real WMRs slip significantly, especially in low-traction, high-speed, or steep conditions. Few attempt to

predictively account for slip in WMR control systems. Most just try to estimate it (e.g. [8][121]) and rely on feedback to correct for it (e.g. [45][80]) but this has disadvantages: slip-induced errors can only be corrected after they occur, and because most WMRs are underactuated it may be difficult to get back on course. Effective model predictive planning and control requires accurate wheel slip prediction. Likewise, predictions should account for real actuator capabilities and not assume any arbitrary sequence of velocity commands is feasible.

2.1.2 Related Work on WMR Dynamics

2D dynamics. There are fewer publications about WMR dynamics than kinematics, and very few about *general* dynamics formulations. Muir's dissertation presented a method to derive WMR dynamics using a force/torque propagation technique based on equating instantaneous power between two coordinate systems [88]; however, like his kinematics work it was limited to 2D models.

Some studied the nonholonomy of 2D mobile robot dynamics. Saha and Angeles [105] used the orthogonal complement to derive constraint-free 2D dynamics, just as they did kinematics. D'Andréa-Novel et al. [28] derived constrained dynamics using Lagrange formalism for a steerable three-wheel vehicle. De Luca and Oriolo [29] studied the dynamics of nonholonomic systems in general, with the example of a planar n-trailer WMR.

Others derived 2D dynamic models for particular platforms, but provide no general method. Zhao and Bement [139] modeled the dynamics of synchronous drive systems (in which wheels are linked by belts or chains) and proved their controllability. Balakrishna and Ghosal [7] and Song and Byun [120] modeled the dynamics of mobile robots with omni wheels and continuously variable transmissions. Yu et al. [137] modeled four-wheel skid-steer vehicles.

Some have modeled planar differential drive robots allowing for nonzero wheel slip. Sidek and Sarkar [115] account for lateral slip only, and Tian et al. [128] account for both lateral and longitudinal slip. Both use the Pacejka "magic formula" tire model. Nandy et al. [90] published a related approach that switches between slip and no-slip domains. These approaches are not generalized to overconstrained systems. Some automotive publications also include a simplified suspension model, i.e. 1-DOF roll dynamics, to better predict handling [119][14]. They do not, however, model the displacement of individual joints, or forces at individual wheels.

3D dynamics. Few model WMR dynamics in full 3D. The equations are typically too cumbersome to derive symbolically, so a numerical "physics engine" is used instead. The closest related work to meeting all the formulation requirements in Section 1.1.1 is the Rover, Analysis, Modeling, and Simulation (ROAMS) software developed at the NASA Jet Propulsion Laboratories (JPL), which modeled the full 3D dynamics of the Mars Exploration Rovers. The intent of Jain et al. [55] and other ROAMS publications is to describe the system rather than provide general WMR modeling methods. The software itself is not publicly available or easily reproducible, though the underlying mathematics is generalizable; ROAMS is based on Spatial Operator Algebra (SOA), a rigid body dynamics formulation developed by Jain and others that can yield $O(n)$ algorithms [100]. Sohl and Jain [118] and Huntsberger et al. [48] present the wheel-ground contact modeling in ROAMS. Contact forces (normal and tangential) are computed via *static* analysis. To make the problem statically determinate, 3-DOF compliance systems are added at each wheel-ground contact point. The Terzaghi equation is used to compute limits on traction

force; but more complex terramechanics are forgone to meet real-time computation requirements [118].

Like ROAMS, Lamon [74] also developed a quasi-static model of the SOLERO platform for a slip minimization controller. Quasi-static models may suffice for slow-moving rovers, but not for high-speed WMRs. Also like ROAMS, Lamon does not use a full terramechanics model, but rather estimates the current rolling resistance online. This is acceptable for an instantaneous controller, but such a model may not be *predictively* accurate enough for MPP.

Others have developed general purpose physics simulators for space robotics which, like ROAMS, are described in publications only at the system level. Examples include SpaceDyn in [136] and Lunar Rover, Modeling, Analysis, and Simulation (LRMAS) in [30]. Due to its availability and simple API, many have used Open Dynamics Engine (OpenDE) to simulate WMR dynamics, such as [52][75][44][101][32][42]. Many also use commercial products like CarSim, Adams/Car, or ANVEL [101]. CarSim and Adams/Car can model automobiles (i.e. Ackerman-steered cars, trucks) in detail, but not other vehicle types. Furthermore, the underlying mathematics/algorithms of these products are proprietary and not well-documented.

Shortcomings. No related work provides a *general* formulation of WMR dynamics. Several software tools exist for dynamic vehicle simulation, but these are either of limited access (e.g. ROAMS), application (e.g. CarSim, Adams/Car), or fidelity (e.g. Open Dynamics Engine).

Many in related work handle wheel-ground contact forces by adding them directly at the contact points [7][52][86]. This method is modular (contact forces may be computed using any model) but many have noted that it makes the dynamics very stiff [7]. In ROAMS, contact forces computed via static analysis are added directly. Sohl and Jain note that, for the compliance systems, a high spring stiffness in the normal direction is desirable to prevent excessive penetration of the wheels into the terrain; but, this causes numerical stiffness which can necessitate an appropriate (more expensive) integration algorithm.

Instead of adding wheel-ground contact forces directly, Open Dynamics Engine includes them implicitly via constraints. OpenDE's implicit method is based on work by Stewart and Trinkle [123]; it is stable but only enforces an approximation of Coulomb's law of friction (see Section 2.4.3). Real wheel-terrain interaction is much more complex.

Motion planning. Several WMR motion planning algorithms have been developed that incorporate kinodynamic constraints, i.e. that require the plan to avoid obstacles and obey the differential constraints of a physically-based dynamical model. To date, however, the WMR models used to demonstrate these algorithms have been limited in either fidelity or speed.

In their motion planning framework for a planetary rover, Iagnemma et al. [51] determined terrain traversability by checking if a feasible solution to a static force balance existed, with terramechanics-based traction limits at the wheels. Eathakota et al. [34] incorporated quasi-static equilibrium and friction cone constraints into their Rapidly-exploring Random Tree (RRT) planning algorithm. But, *static* analysis is insufficient for high-speed applications. LaValle and Kuffner [76] demonstrated RRT motion planning with *dynamic* models, but not of WMRs; they planned trajectories for spacecraft with contact-free dynamics, controlled by direct force/torque inputs at the cg. Pivtoraiko and Kelly [96] demonstrated their kinodynamic lattice planning algorithm using an Open Dynamics Engine model of a tricycle on slippery, but flat ground. Howard demonstrated MPC/MPP algorithms with 3D kinematic models on rough terrain, but deemed dynamic models to be too expensive [46][47].

Ishigami et al. [53] presented a planning algorithm that performed full dynamic simulation of candidate paths, including terramechanics-based wheel-terrain interaction, but the computational burden was high. Simulating just four paths (that would each require only 3 minutes for the robot to execute) took 47 minutes. Tools like CarSim are useful for design evaluation, but are not fast enough for model predictive planning; CarSim runs at best $10\times$ faster than real-time on 3 GHz processor [87].

In summary, many planning algorithms are available that can utilize a dynamic WMR model, but constructing models of sufficient fidelity and speed remains a challenge.

2.1.3 Comparison of Related Work on Model Formulation

Table 2.1 categorizes the capabilities of related work and this thesis. The columns are:

- Publication group: Rows are ordered chronologically by the earliest publication in each group.
- Mathematical basis: as explained in the introduction of Section 2.1.
- Kinematics
- Dynamics
- 3D: vs. planar only
- General: provides a systematic way to model any joint configuration
- Constrained simulation: uses a DAE to solve directly for motions that respect constraints
- Wheel slip prediction: can predict nonzero wheel slip
- Experiment: validates the model in physical experiments

Table 2.1 alone does not fully convey our contribution. While others have provided capabilities like constrained simulation or wheel slip prediction, we do so in a novel way. We also provide capabilities not covered in the table like modularity.

Table 2.1: Comparison of related work on WMR model formulation

Publication group	Basis	Kin.	Dyn.	3D	General	Const. sim.	Slip pred.	Exp.
Muir and Neuman 1986, Muir 1988: *first 2D general methods*	transform	X	X		X			
Agullo et al. 1987, Burke and Durrant-Whyte 1993, Kanayama 1994, Borenstein 1995: *modeled particular platforms*	geometry	X						
Alexander and Maddocks 1989, Rajagopalan 1997, Shin and Park 2001: *extended Muir and Neuman*	transform	X			X			
Saha and Angeles 1989, D'Andréa-Novel et al. 1991: *derived Lagrange constrained 2D dynamics, no wheel slip*	twist, geometry	X	X					
Balakrishna and Ghosal 1995, Song and Byun 2004, Yu et al. 2010: *modeled particular platforms*	geometry	X	X					
Yoshida 1999, Ding et al. 2008: *software for space robotics simulation, used recursive dynamics algorithms*	various	X	X	X				
Choi and Sreenivasan 1999, Chakraborty and Ghosal 2004, Eathakota et al. 2011, Gattupalli et al. 2013: *modeled particular platforms in 3D using DAEs*	geometry, adjoint	X		X		X		
Balaram 2000, Lamon and Siegwart 2007, Seegmiller and Wettergreen 2011: *used 3D kinematic models in experiments, no general methods*	geometry, adjoint	X		X				X
Jain et al. 2003, Sohl and Jain 2005, Huntsberger et al. 2008: *ROAMS*	SOA	X	X	X			X	X
Kim et al. 2004, Fu and Krovi 2008: *alternative 2D general kinematics methods*	transfer, twist	X			X			
Tarokh and McDermott 2005: *first 3D general kinematics method*	transform	X		X	X			
Le Menn et al. 2006, Benamar et al. 2010: *modeled multi-monocycle robots*	screw	X		X				
Tarokh and Mcdermott 2007, Tarokh et al. 2013, Chang et al. 2006, 2009: *3D general kinematics methods, velocity propagation-based*	velocity prop.	X		X	X			
Sidek and Sarkar 2008, Tian et al. 2009, Nandy et al. 2011: *derived 2D diff. drive dynamics with wheel slip prediction*	geometry		X				X	
Kelly 2010, Kelly and Seegmiller 2012: *coordinate-free method based on Transport Theorem*	velocity prop.	X		X	X	X		X
Seegmiller and Kelly 2013, 2014: *this thesis*	SVA	X	X	X	X	X	X	X

Major theoretical contributions of this thesis to WMR model formulation include the following:

- This work is the first to combine a general derivation of WMR kinematics with the use of DAEs and Baumgarte's stabilization method for constrained, drift-free motion prediction (Section 2.2). This contribution was made in collaboration with Alonzo Kelly, building on his prior work [62]. All remaining contributions were made independently by the author.
- This thesis presents a novel enhancement to the kinematic model, in which nonzero wheel slip is predicted in a principled way based on gravitational, inertial, and dissipative forces. In practice, the enhanced kinematic model can provide comparable accuracy to a full dynamic model at a fraction of the computational cost. (Section 2.3)
- This thesis provides the first general method for the construction of 3D dynamic WMR models. The closest related work only considers kinematics and does not predict wheel slip [127][25]. (Section 2.4)
- For our dynamics formulation, we developed a novel method of enforcing realistic wheel-terrain interaction models. We use "force-balance optimization" to enforce constraints in a DAE formulation, which permits larger integration steps than the conventional method of adding forces directly. Some alternatives like Open Dynamics Engine also use constraints, but can only approximate Coulomb friction at contacts. In contrast, we can enforce any contact model, including nonlinear empirical and terramechanics-based models. The method also supports nonlinear actuator models. (Section 2.4.3)

More detailed distinctions between this and related work will be made in later sections explaining our formulation. Additional contributions include: adaptation of SVA algorithms for use in WMR modeling, resolution of overconstraint and nonlinearity issues, techniques for faster computation, and extensive experimental validation. We have several publications on this work: [64][66] present the use of DAEs and stabilization for kinematic motion prediction, [111] presents our enhanced kinematics formulation, and [110] presents our dynamics formulation.

2.2 Kinematic Model Formulation

This section explains our kinematic model formulation. First, Section 2.2.1 describes the specification of WMR frame information and state space. Next, Sections 2.2.2 and 2.2.3 explain the formation of wheel-ground contact and joint constraints. Section 2.2.4 presents an efficient method to initialize wheel-terrain contact using these constraints. Finally, Section 2.2.5 presents our "stabilized DAE" method for predicting motions that respect these constraints.

2.2.1 Specification of Model Information

Frame information. Model information is specified as an ordered list of frames, which are connected in a kinematic tree. The root of the tree is the body frame (b), which has 6 degrees of freedom (DOF) with respect to the ground-fixed navigation/world frame (w). All other frames have one DOF with respect to their parent frame. Higher DOF joints (such as a 2-DOF universal joint) can be modeled with multiple coincident frames. All frames use right-handed coordinate

systems, and the body frame uses the coordinate system convention from the ISO 8855 standard: x-forward, y-left, z-up [54].

Unlike some related work [127][25], our formulation is compatible with, but does not require use of the Denavit-Hartenberg (D-H) convention. Adherence to the D-H convention restricts the location of frame origins, sometimes to non-intuitive locations away from the physical joint [136]. Unlike Open Dynamics Engine, we do not require collocation of the origin and center of mass.

To illustrate the model specification process, Figure 2.2 and Table 2.2 present the frame information for the Rocky 7 rover. The columns of Table 2.2 are:

- col 1: index. Frames are ordered such that every frame's index is greater than its parent's: $i > p(i)$.
- col 2: frame name (bold font for wheel frames)
- col 3: parent frame name
- col 4: joint type: R/P for revolute/prismatic about the X/Y/Z axis
- col 5: actuation: Y/N for yes/no
- cols 6-11: Pose of the of the frame with respect to its parent frame when joint displacement is zero ($\boldsymbol{\rho}_i^{p(i)}|\theta = 0$). Pose comprises translation $[x\ y\ z]^T$ and orientation $[\theta_x\ \theta_y\ \theta_z]^T$ in Euler angles.

Dynamic models also require that the mass, center of mass, and moment of inertia be specified. In our software implementation, specifying this information requires just 1-2 lines of code per frame.

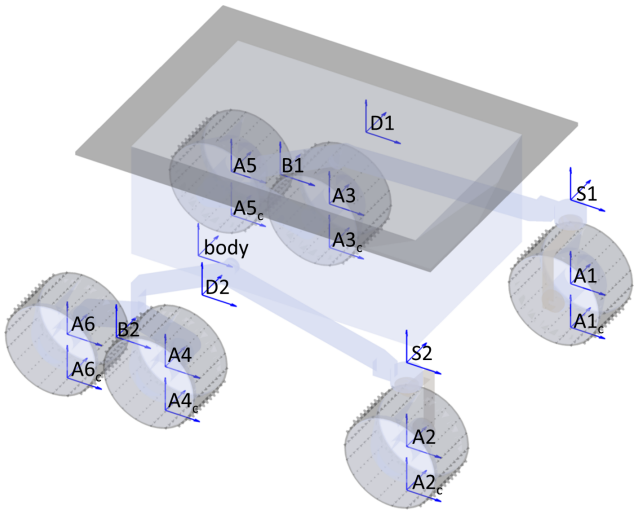


Figure 2.2: Rocky 7 frame diagram

Contact frames. All branches of the kinematic tree terminate with wheel frames, which by convention attach via revolute joints about their y axes. Special contact frames are appended to each wheel frame, defined such that their origin is coincident with the wheel-ground contact point. In reality contact occurs over one or more patches on the wheel surface, but we approximate to a single point. The contact frame z-axis is aligned with the terrain normal, and the x

Table 2.2: Rocky 7 frame information

i	Frame	Parent	Type	Act.	x	y	z	θ_x	θ_y	θ_z
1	body (b)	world (w)								
2	D1	body	RY	N	k2	k3	k1	0	0	0
3	S1	D1	RZ	Y	k4	0	0	0	0	0
4	**A1**	S1	RY	Y	0	0	-k5	0	0	0
5	B1	D1	RY	N	-k6x	0	-k6z	0	0	0
6	**A3**	B1	RY	Y	k7	0	-k8	0	0	0
7	**A5**	B1	RY	Y	-k7	0	-k8	0	0	0
8	D2	body	RY	N	k2	-k3	k1	0	0	0
9	S2	D2	RZ	Y	k4	0	0	0	0	0
10	**A2**	S2	RY	Y	0	0	-k5	0	0	0
11	B2	D2	RY	N	-k6x	0	-k6z	0	0	0
12	**A4**	B1	RY	Y	k7	0	-k8	0	0	0
13	**A6**	B1	RY	Y	-k7	0	-k8	0	0	0

in centimeters: k1=10.5, k2=12.075, k3=20, k4=28.8, k5=12.5, k6x=16·sin(49°), k6z=16·cos(49°), k7=6.9, k8=2, wheel radius=6.5, total mass = 11 kg

and y axes are aligned with the longitudinal and lateral slip directions (see Figure 2.3). Frames named with subscript "c" in Figure 2.2 are the Rocky 7 rover's contact frames.

In a simulation/prediction context, contact points are determined via collision detection between wheel and terrain geometries. Wheels can be represented as 3D circles, and terrain as a mesh or height map generated by perception. Collision detection also outputs the "contact height error" (Δz) for each point; this is the difference in height between the contact point (on the wheel) and terrain surface, measured along the surface normal:

$$\Delta z = (\vec{r}_{\textit{contact}} - \vec{r}_{\textit{terrain}}) \cdot \hat{n} \tag{2.8}$$

The vectors in (2.8) are defined as follows (in world coordinates):

$$\begin{aligned} \vec{r}_{\textit{contact}} &= [x,\ y,\ z]^T \\ \vec{r}_{\textit{terrain}} &= [x,\ y,\ \text{terrainHeight}(x,y)]^T \\ \hat{n} &= \text{terrainNormal}(x,y) \end{aligned}$$

The contact point ($\vec{r}_{\textit{contact}}$) is chosen as the point on the wheel surface for which Δz is smallest (or most negative). A positive Δz indicates separation, and a negative Δz indicates penetration (see Figure 2.4). We assume that the terrain height and normal vector can be computed at any x,y location.

Contact angles (δ) are sometimes used to specify the contact point location. We define δ such that:

$$\vec{r}_{\textit{contact}} = \vec{r}_{\textit{wheel}} - R(\sin(\delta)\hat{x} + \cos(\delta)\hat{z}) \tag{2.9}$$

where R is the wheel radius, and $\hat{x}$/$\hat{z}$ are the wheel frame x/z axis unit vectors. The contact angle is depicted in Figure 2.3. This angle can often be sensed or estimated, and the terrain normal can be approximated as pointing toward the wheel frame origin.

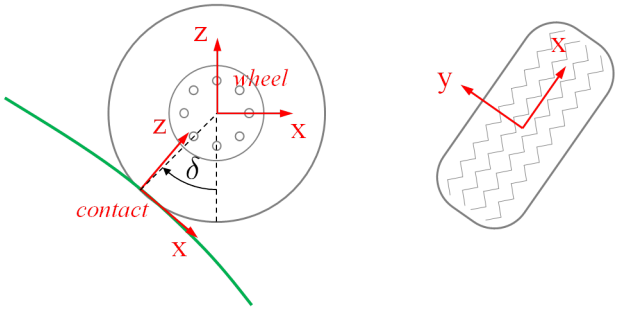


Figure 2.3: Contact frame diagram. Contact frame axis directions are: x-longitudinal, y-lateral, z-terrain normal. The contact angle (δ) specifies the location of the wheel-ground contact point. Specifically, δ represents a counter-clockwise rotation about the wheel frame's y axis, starting from its -z axis.

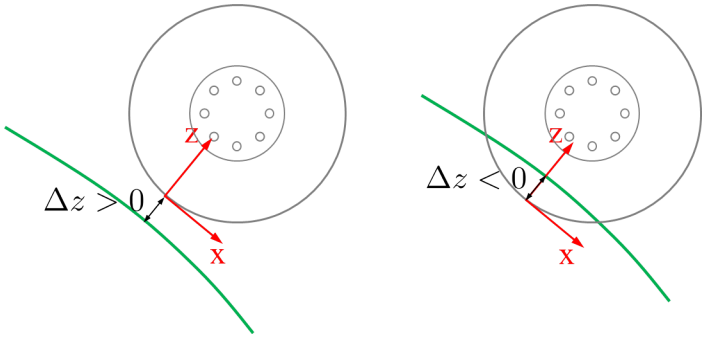


Figure 2.4: Contact height error diagram. Positive Δz indicates separation, negative indicates penetration.

State vector. We use generalized or reduced coordinates to compactly represent the WMR state in our model formulations. The state vector is:

$$\mathbf{q} = \begin{bmatrix} \boldsymbol{\rho}_b^w \\ \boldsymbol{\theta} \end{bmatrix}, \quad \boldsymbol{\rho}_b^w = \begin{bmatrix} \mathbf{o}_b^w \\ {}^{w}\vec{r}_b^{\,w} \end{bmatrix} \tag{2.10}$$

The first subvector is the pose ($\boldsymbol{\rho}$) of the body frame with respect to the world frame, which comprises orientation ($\mathbf{o}$, expressed using Euler angles or a quaternion) and position ($\vec{r}$). The remaining elements are the revolute/prismatic joint displacements ($\boldsymbol{\theta}$).

Our state space is smaller than the "maximal coordinates" space used in some Lagrange multiplier-based dynamics formulations, such as Baraff [9] and Open Dynamics Engine. The maximal coordinates space contains the 6-DOF pose of each frame, which necessitates numerous constraint equations for lower pairs; each 1-DOF revolute or prismatic joint requires a 5-DOF constraint. Big matrices must be inverted at every time step to solve for the Lagrange multipliers, but computational cost can be mitigated somewhat by exploiting sparsity. Reduced coordinates avoid these difficulties by satisfying joint constraints implicitly.

Given the frame information and $\mathbf{q}$, we can readily solve the forward position kinematics. Specifically, we can compute the homogeneous transform between each frame i and its parent frame $p(i)$:

$$T_i^{p(i)} = \begin{bmatrix} R_i^{p(i)} & {}^{p(i)}\vec{r}_i^{\,p(i)} \\ [0] & 1 \end{bmatrix} \tag{2.11}$$

Rotation (R) depends on the joint displacement (θ) for revolute joints, whereas translation ($\vec{r}$) depends on the joint displacement for prismatic joints.

Algorithm 1 details the calculation of these homogeneous transforms. The transform from body (i.e. frame 1) to world coordinates is computed from the pose vector in $\mathbf{q}$ (line 1). The transform to parent coordinates if joint displacement is zero (line 4) can be precomputed for each joint and stored with the frame information. n_f is the number of user-defined frames in the WMR model, which does not include contact frames. $a(i)$ denotes the axis number for joint i (x=1, y=2, z=3), and $\mathrm{Rot}(\textit{axis number}, \textit{angle})$ denotes an axis-angle specified rotation matrix. The algorithm also computes the homogeneous transform of each frame with respect to the world frame (line 12). The pose of the wheel frames in world coordinates are required for collision detection.

Algorithm 1 Forward position kinematics

1: $T_b^w \leftarrow \mathrm{poseToTransform}(\boldsymbol{\rho}_b^w)$
2: **for** $i = 2$ to n_f **do**
3: $\quad\theta \leftarrow \boldsymbol{\theta}(i-1)$
4: $\quad T_0 \leftarrow \mathrm{poseToTransform}(\boldsymbol{\rho}_i^{p(i)} | \theta = 0)$
5: $\quad R_0 \leftarrow T_0(1\ldots3, 1\ldots3)$
6: $\quad \vec{r}_0 \leftarrow T_0(1\ldots3, 4)$
7: $\quad$ **if** joint i type = revolute **then**
8: $\quad\quad T_i^{p(i)}(1\ldots3, 1\ldots3) \leftarrow R_0 \,\mathrm{Rot}(a(i), \theta)$
9: $\quad$ **else if** joint i type = prismatic **then**
10: $\quad\quad T_i^{p(i)}(1\ldots3, 4) \leftarrow \vec{r}_0 + R_0(a(i), 1\ldots3)^T \cdot \theta$
11: $\quad$ **end if**
12: $\quad T_i^w \leftarrow T_{p(i)}^w T_i^{p(i)}$
13: **end for**

Joint space velocity. To simplify equations in the kinematics and dynamics, we define an alternative to the time derivative of state ($\dot{\mathbf{q}}$) called the joint space velocity ($\dot{\mathbf{q}}'$):

$$\dot{\mathbf{q}} = \frac{d}{dt}(\mathbf{q}) = \begin{bmatrix} \dot{\boldsymbol{\rho}}_b^w \\ \dot{\boldsymbol{\theta}} \end{bmatrix} \tag{2.12}$$

$$\dot{\mathbf{q}}' = \begin{bmatrix} {}^{b}\vec{v}_b^{\,w} \\ \dot{\boldsymbol{\theta}} \end{bmatrix} \tag{2.13}$$

The time derivative of the body frame pose in $\dot{\mathbf{q}}$ is replaced with the spatial velocity of the body frame (in body coordinates) in $\dot{\mathbf{q}}'$. Conversion between the two requires multiplication by a block-diagonal transform:

$$\dot{\mathbf{q}} = V(\mathbf{q})\dot{\mathbf{q}}' \tag{2.14}$$

$$\frac{d}{dt}\begin{bmatrix} \mathbf{o}_b^w \\ {}^{w}\vec{r}_b^{\,w} \\ \boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \Omega & & \\ & R_b^w & \\ & & I \end{bmatrix} \begin{bmatrix} {}^{b}\vec{\omega}_b^{\,w} \\ {}^{b}\vec{v}_b^{\,w} \\ \dot{\boldsymbol{\theta}} \end{bmatrix}$$

Appendix A explains the conversion of orientation vector rates to angular velocity via the Ω matrix. Note that Ω is a function of the orientation vector $\mathbf{o}_b^w$, and the rotation matrix R_b^w is an equivalent representation of it.

In our dynamics formulation (Section 2.4) we also require the time derivative of joint space velocity, $\ddot{\mathbf{q}}'$ or the joint space acceleration:

$$\ddot{\mathbf{q}}' = \frac{d}{dt}(\dot{\mathbf{q}}') = \begin{bmatrix} {}^{b}\vec{a}_b^{w} \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} \tag{2.15}$$

2.2.2 Wheel-Ground Contact Constraints

We predict WMR motion by enforcing constraints on the relative motion between the wheels and the ground. Each wheel in contact with the ground provides a three-row constraint of the form:

$$A_{\textit{wheel}}\dot{\mathbf{q}}' = \vec{v}_c \tag{2.16}$$

Where $\vec{v}_c$ is the velocity of the contact point with respect to the ground in local coordinates $({}^{c}\vec{v}_c^{w})$. We will refer to $A_{\textit{wheel}}$ as the "wheel Jacobian."

The first two rows of (2.16) are nonholonomic constraints restricting velocities in the longitudinal (x) and lateral (y) slip directions. The third row is the *time derivative* of the holonomic constraint, restricting velocity in the surface normal direction (z).

The Jacobian $A_{\textit{wheel}}$ is computed using a vector algebra relation derived from the *Transport Theorem*. As explained in Kelly and Seegmiller [64], the Transport Theorem relates observations of motion in moving frames:

$$\left.\frac{d\vec{u}}{dt}\right|_f = \left.\frac{d\vec{u}}{dt}\right|_m + \vec{\omega}_m^f \times \vec{u} \tag{2.17}$$

$\vec{u}$ in (2.17) can represent position or any of its time derivatives (velocity, acceleration, etc.). f and m denote a fixed and moving observer respectively (in practice both may be moving). Due to their relative motion, the two observers would compute different time derivatives for the same vector $\vec{u}$.

Substituting position ($\vec{r}$) for $\vec{u}$ yields this velocity relation:

$$\vec{v}_o^f = \vec{v}_m^f + \vec{v}_o^m + \vec{\omega}_m^f \times \vec{r}_o^m \tag{2.18}$$

The variables in (2.18) are depicted in Figure 2.5. The linear velocity of object o in reference frame f can be computed from its position and linear velocity in frame m, if the relative linear and angular velocities between the frames are known. Though not required here, the Transport Theorem can also be used to derive acceleration and higher-order kinematics [65].

We use (2.18) to propagate velocities up the kinematic chain for each contact point. These chains can be extracted from the kinematic tree by starting at the contact frame (c) and adding successive parent frames. For example, the kinematic chain for the Rocky 7 rover's front right wheel is:

$$w \leftarrow b \leftarrow D2 \leftarrow S2 \leftarrow A2 \leftarrow A2_c \tag{2.19}$$

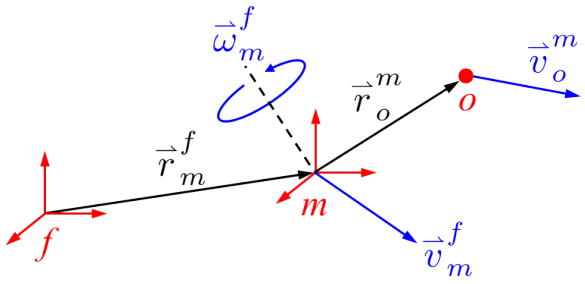


Figure 2.5: Variables in the velocity relation derived from the Transport Theorem (2.18)

Position, linear velocity, and angular velocity can be propagated up this chain as follows:

$$\vec{r}^{\,w}_{i+1} = \vec{r}^{\,w}_{i} + \vec{r}^{\,i}_{i+1} \tag{2.20}$$
$$\vec{v}^{\,w}_{i+1} = \vec{v}^{\,w}_{i} + \vec{v}^{\,i}_{i+1} + \vec{w}^{\,w}_{i} \times \vec{r}^{\,i}_{i+1} \tag{2.21}$$
$$\vec{\omega}^{\,w}_{i+1} = \vec{\omega}^{\,w}_{i} + \vec{\omega}^{\,i}_{i+1} \tag{2.22}$$

where i is the index *in the chain*, not in the frame list for the entire WMR.

Let 0 and n be the indices of the world and contact frames in the chain, respectively. Then, based on the above propagation equations, the velocity of the contact point with respect to the ground-fixed world frame can be expressed with summations:

$$\vec{v}^{\,0}_{n} = \sum_{i=0}^{n-1} \vec{v}^{\,i}_{i+1} + \sum_{i=0}^{n-2} \vec{\omega}^{\,i}_{i+1} \times \vec{r}^{\,i+1}_{n} \tag{2.23}$$

This expression makes the contribution of each joint clear. $\vec{\omega}^{\,i}_{i+1}$ is nonzero for revolute joints, $\vec{v}^{\,i}_{i+1}$ is nonzero for prismatic joints, and both are nonzero for the fictitious 6-DOF joint between the world and body frames (0 and 1).

Algorithm 2 details the wheel Jacobian calculation for a single wheel. This algorithm proceeds directly from (2.23); it loops through the kinematic chain from contact to body frame accounting for each joint. Here, i is the index in the frame list for the entire WMR (not in a particular chain). On lines 6 and 8, the rotation matrix columns $R^{w}_{i}(*, a(i))$ are the axes along which the revolute and prismatic joint velocities are directed (i.e. the unit vectors $\hat{\omega}^{p(i)}_{i}$, $\hat{v}^{p(i)}_{i}$). Line 12 accounts for the angular velocity of the body frame, and uses the identity: $\vec{\omega}\times\vec{r} = -\vec{r}\times\vec{\omega} = [\vec{r}]^{T}_{\times}\vec{\omega}$.

Note that the Transport Theorem yields a coordinate-free velocity relation (2.18) which allows flexibility in the wheel Jacobian calculation. In Algorithm 2, we leverage the homogeneous transforms between each frame and the world frame (T^{w}_{i}) already computed for collision detection. All rotation matrices (R) and position vectors ($\vec{r}$) are obtained directly from these transforms. The final step (line 14) transforms from world to contact frame coordinates such that nonholonomic (x,y) and holonomic (z) constraints are separated.

Algorithm 2 Wheel Jacobian calculation (for a single wheel)

$A_{wheel} \leftarrow [0]$
$c \leftarrow$ contact frame index
$i \leftarrow p(c)$
while $i > 1$ **do**
    if joint i type = revolute **then**
        $A_{wheel}(*, i+5) \leftarrow R_i^w(*, a(i)) \times (\vec{r}_c^{\,w} - \vec{r}_i^{\,w})$
    else if joint i type = prismatic **then**
        $A_{wheel}(*, i+5) \leftarrow R_i^w(*, a(i))$
    end if
    $i \leftarrow p(i)$
end while
$A_{wheel}(*, 1...3) \leftarrow [\vec{r}_c^{\,w} - \vec{r}_1^{\,w}]_\times^T R_1^w$
$A_{wheel}(*, 4...6) \leftarrow R_1^w$
$A_{wheel} = (R_c^w)^T A_{wheel}$

Constraint equations for all wheels can be stacked into a single matrix equation:

$$A_{wheel}\dot{\mathbf{q}}' = \mathbf{v}_c \tag{2.24}$$
$$\begin{bmatrix} A_{wheel,1} \\ \vdots \\ A_{wheel,n_w} \end{bmatrix} \dot{\mathbf{q}}' = \begin{bmatrix} \vec{v}_{c,1} \\ \vdots \\ \vec{v}_{c,n_w} \end{bmatrix}$$

Hereafter let A_{wheel} refer to the stacked matrix of wheel Jacobians with three rows per each wheel in contact with the ground.

2.2.3 Holonomic Joint Constraints

Many articulated platforms have additional holonomic joint constraints. For example, a mechanical linkage constrains the Rocky 7 rover's two rocker angles to be symmetric. Mathematically this constraint can be expressed:

$$c = \mathbf{q}(\text{D1}) + \mathbf{q}(\text{D2}) = 0 \tag{2.25}$$

Where D1 and D2 are the indices of the rocker joint angles in the state vector. The time derivative of this constraint is that the sum of the rocker joint rates be zero:

$$A_{joint}\dot{\mathbf{q}}' = 0 \tag{2.26}$$
$$A_{joint} = [[0\ 0\ 0\ 0\ 0\ 0]\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$$

The rocker frame indices are 2 and 8 in the frame list (Table 2.2) and thus their rates are indices 7 and 13 in the joint space velocity.

2.2.4 Initialization of Terrain Contact

Here we present an efficient method to initialize state ($\mathbf{q}$) such that wheels contact the terrain surface. Many have required this "terrain fitting" procedure [51][125][46], but performed it differently than presented here. We require this procedure to initialize simulations, but do not perform it at every time step like [125] (see Section 2.2.5).

Assume that some elements of $\mathbf{q}$ are fixed and some are free. Fixed elements typically include x, y, yaw, and actuated joint displacements. Free elements typically include z, roll, pitch, and passive joint displacements. Let $\mathbf{x}$ denote the subvector of free states in $\mathbf{q}$ to be solved for:

$$\mathbf{x} = \mathbf{q}(\text{free}) \tag{2.27}$$

The optimal values for these free states ($\mathbf{x}^*$) minimize the contact height error (Δz, defined in (2.8)) for all wheels:

$$\begin{aligned} \mathbf{x}^* &= \arg\min_{\mathbf{x}} L(\mathbf{x}) \\ L(\mathbf{x}) &= \|\mathbf{e}(\mathbf{x})\|^2 = \mathbf{e}(\mathbf{x})^T\mathbf{e}(\mathbf{x}) \\ \mathbf{e}(\mathbf{x}) &= [\Delta z_1 \ \dots \ \Delta z_{n_w}]^T \end{aligned} \tag{2.28}$$

where n_w is the number of wheels.

Whereas [125] used a derivative-free minimization algorithm, we utilize wheel Jacobians for faster convergence. Efficient, robust minimization is possible using Newton's method combined with a line search algorithm; more detail is provided in Section 2.4.3 on force-balance optimization.

Minimization of $L(\mathbf{x})$ via Newton's method requires the Jacobian of $\Delta\mathbf{z}$ with respect to $\mathbf{x}$ to compute the gradient and Hessian. This can be computed from the wheel Jacobians:

$$\frac{\partial \Delta\mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \Delta\mathbf{z}}{\partial \mathbf{q}}(*, \text{free}) \tag{2.29}$$

$$\frac{\partial \Delta\mathbf{z}}{\partial \mathbf{q}} = \frac{\partial \Delta\dot{\mathbf{z}}}{\partial \dot{\mathbf{q}}} \tag{2.30}$$

$$\frac{\partial \Delta\dot{\mathbf{z}}}{\partial \dot{\mathbf{q}}} = \frac{\partial \Delta\dot{\mathbf{z}}}{\partial \dot{\mathbf{q}}'} V(\mathbf{q})^T \tag{2.31}$$

$$\frac{\partial \Delta\dot{\mathbf{z}}}{\partial \dot{\mathbf{q}}'} = \frac{\partial \mathbf{v}_c(\mathrm{z})}{\partial \dot{\mathbf{q}}'} = A_{\mathit{wheel}}(\mathrm{z}, *) \tag{2.32}$$

The set z in (2.32) contains the indices of constraints on contact point velocities in the z-axis (or terrain normal) direction; i.e. $A_{\mathit{wheel}}(\mathrm{z}, *)$ comprises every third row of A_{wheel}. The matrix V is defined in (2.14). This Jacobian calculation does not account for changes in contact angle or terrain height, but it is an effective approximation in practice.

If applicable, holonomic joint constraints (as explained in Section 2.2.3) can be appended to the objective function so they too are enforced when initializing terrain contact.

2.2.5 Kinematic Motion Prediction

Here we explain the process of predicting motions that respect kinematic constraints. The combined constraint equation is:

$$A(\mathbf{q})\dot{\mathbf{q}}' = \mathbf{v} \tag{2.33}$$

This equation comprises all wheel constraints (2.24) and any additional holonomic joint constraints (2.26):

$$\begin{bmatrix} A_{wheel} \\ A_{joint} \end{bmatrix} \dot{\mathbf{q}}' = \begin{bmatrix} \mathbf{v}_c \\ \mathbf{v}_{joint} \end{bmatrix} \tag{2.34}$$

$\mathbf{v}_c$ on the right hand side comprises the contact point velocities for each wheel ($\vec{v}_c$). Kinematic simulations typically assume no wheel slip, so the x and y components of $\vec{v}_c$ are set to zero. Setting the z component to zero would only enforce the holonomic contact constraint to first order; eventually, drift would cause the wheels to either penetrate or separate from the terrain surface. We use Baumgarte's stabilization method to counteract this drift [10]. This means setting the z component of $\vec{v}_c$ proportional to the contact height error, such that the error is driven to zero:

$$\vec{v}_c = \begin{bmatrix} 0 \\ 0 \\ -\sigma\Delta z \end{bmatrix} \tag{2.35}$$

where σ is a proportional gain bounded by: $0 < \sigma \leq \frac{1}{\Delta t}$. Baumgarte's stabilization method is also used for holonomic joint constraints, i.e. (2.26) becomes:

$$A_{joint}\dot{\mathbf{q}}' = -\sigma c \tag{2.36}$$

We allow some elements of $\dot{\mathbf{q}}'$ in (2.33) to be known and move them to the right hand side:

$$A(*, \text{unknown})\dot{\mathbf{q}}'(\text{unknown}) = \mathbf{v} - A(*, \text{known})\dot{\mathbf{q}}'(\text{known}) \tag{2.37}$$

where known and unknown are complementary subsets of the set of all indices.

In a simulation/prediction context, the known elements of $\dot{\mathbf{q}}'$ are commanded rates for actuated joints. In an estimation context, the known elements are sensed joint rates (via encoders) and possibly the sensed angular velocity of the body frame (via gyroscope).

(2.37) can also be used in a control context, in which the known elements are the desired velocity of the body frame and the unknown elements are actuated joint rates. In control, one may also need to solve for joint displacements (e.g. steer angles); A may depend nonlinearly on these values.

The system in (2.37) will typically be overconstrained, and can be solved using the pseuodinverse:

$$\dot{\mathbf{q}}'(\text{unknown}) = A(*, \text{unknown})^{+} \left(\mathbf{v} - A(*, \text{known})\dot{\mathbf{q}}'(\text{known})\right) \tag{2.38}$$

This minimizes the slip velocities at all wheels in a least-squares sense.

2.2.6 Kinematic Simulation Process

This section provides an overview of the simulation process. Note that throughout this document, "simulation" is used synonymously with "prediction" (i.e. for MPC or MPP applications).

The flowchart in Figure 2.6 summarizes the process of forward simulating our kinematic and dynamic models. The user must provide the following: frame information for the WMR (e.g. Table 2.2 for Rocky 7), functions to compute the terrain height and normal at any (x,y) location, a controller function that outputs rate commands for actuated joints, and an initial state vector.

The "forward position kinematics" block is the calculation of homogeneous transforms for user-defined frames (see Algorithm 1). The "collision detection" block is the calculation of homogeneous transforms and height errors for the wheel-ground contact frames (see Section 2.2.1).

The "forward velocity kinematics" block is the calculation of joint space velocity ($\dot{\mathbf{q}}'$) per Section 2.2.5. Finally, the "integration" block is the integration of $\dot{\mathbf{q}}'$ to update state. Using the Euler method:

$$\mathbf{q}[i+1] = \mathbf{q}[i] + V(\mathbf{q}[i])\dot{\mathbf{q}}'[i]\Delta t \tag{2.39}$$

For better accuracy, higher-order methods like Runge-Kutta can be used.

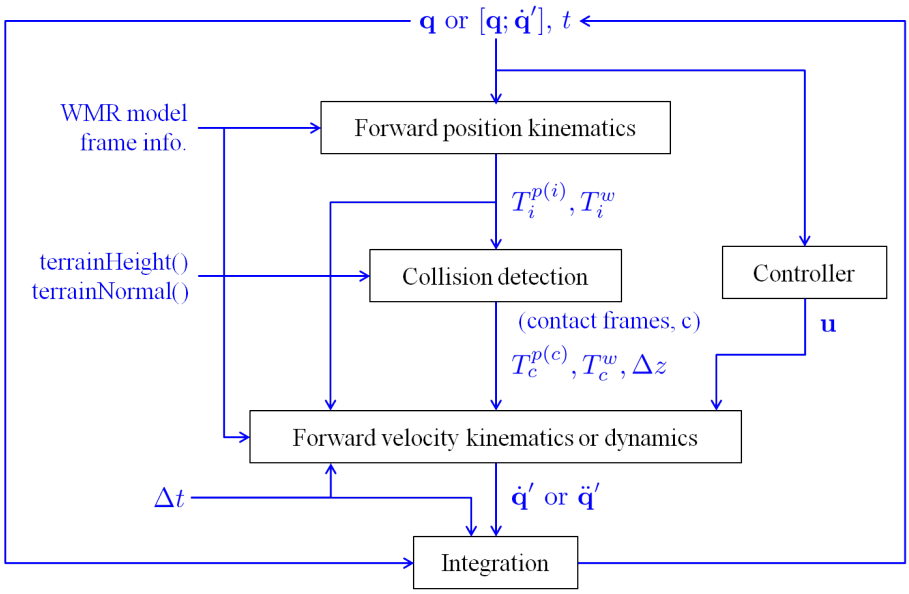


Figure 2.6: Process flowchart for kinematic or dynamic simulation

By solving for constrained motion in a DAE formulation and using Baumgarte's method for stabilization of the holonomic constraints, we eliminate the need for "terrain fitting" to keep wheels on the ground. We compare these approaches in Section 2.5.1.

Limitations. The 3D kinematic models described here are higher-fidelity than those used in most fielded WMRs today, but still have limitations. As in most kinematic models, inertial effects are neglected. In particular, gravity and acceleration can cause liftoff and rollover on real WMRs, but these phenomena are not predicted by the kinematic model. In addition, the kinematic model minimizes slip at the wheels, but real WMRs can slip significantly. To address

these limitations, Section 2.3 presents an enhancement to our kinematic model that enables the prediction of nonzero wheel slip, and the assessment of rollover risk.

2.3 Enhancement of the Kinematic Model

The primary limitation of the kinematic model formulation in Section 2.2 is the unrealistic assumption that wheels do not slip. In fact, no prior work on general methods for WMR kinematic modeling predicts nonzero wheel slip, except when wheels are misaligned and slip is minimized via the pseudoinverse [89][125][64]. Of course real WMRs slip even with their wheels properly aligned, especially in low-traction, high-speed, or steep conditions.

The obvious way to include slip in our kinematic model is to set the x,y components of $\vec{v}_c$ in (2.35) to nonzero values, but to what values? The wheel-ground contact models used in the dynamic formulation (Section 2.4.3) relate slip to force, but the kinematic formulation provides no information about contact forces. Besides being unknown, wheel slip values may be unnecessary; in many applications only the motion of the vehicle *body* matters. Accordingly, we developed a novel enhancement to the kinematics to predict *body-level* slip. These slip predictions are parametrized about gravitational, inertial, and dissipative forces in a principled way.

Empirically, slip ratio (s) and angle (α) are linearly proportional to *normalized* longitudinal and lateral force, up to a limit (see [19], [133] Section 1.3). Normalized means the ratio of longitudinal/lateral force to normal force. Let v_x,v_y denote the x,y components of $\vec{v}_c$ (i.e. slip velocities); let f_x,f_y denote the x,y components of $\vec{f}_c$ (i.e. contact forces); and let V_x,V_y denote the velocities of the wheel center with respect to the ground. Let all variables be expressed in contact frame coordinates. Then:

$$s = \frac{r\omega - V_x}{V_x} = \frac{-v_x}{V_x} \propto \frac{f_x}{f_z} \tag{2.40}$$

$$\therefore v_x \propto \frac{f_x}{f_z} V_x \tag{2.41}$$

$$\alpha = \tan^{-1}\left(\frac{v_y}{V_x}\right) \approx \frac{v_y}{V_x} \propto \frac{f_y}{f_z} \tag{2.42}$$

$$\therefore v_y \propto \frac{f_y}{f_z} V_x \tag{2.43}$$

Up to a limit, slip velocity is linearly proportional to the product of normalized force (in the direction of slip) and longitudinal velocity. We extend this wheel-level observation to model slip at the body-level.

We compute force at the body frame as follows:

$$\vec{f}_b = I_b^c \vec{g} - \vec{v}_b \times_f I_b^c \vec{v}_b \tag{2.44}$$

Note that (2.44) uses spatial vector algebra (SVA) and matches (2.89), the approximate joint space bias force calculation in the dynamics formulation. It accounts for gravity and Coriolis and centripetal acceleration. $\vec{v}_b$ (short for ${}^{b}\vec{v}_b^{w}$) is obtained from the least-squares slip solution for $\dot{\mathbf{q}}'$, computed as explained in Section 2.2.5. All variables are expressed in body coordinates.

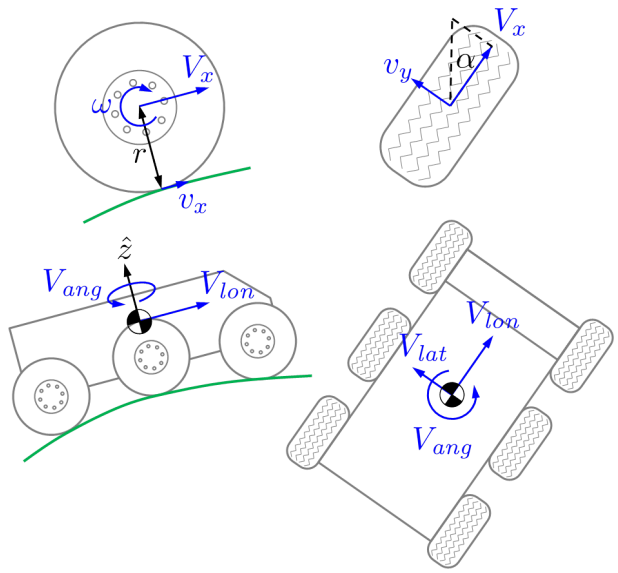


Figure 2.7: Diagram of wheel-level slip variables (top) and body-level slip variables (bottom)

We do not parametrize slip directly about $\vec{f}_b$, because the magnitude of $\vec{f}_b$ depends on where the user chooses to place the body frame, but physical slip does not. For consistency, we transform the body frame quantities to a new frame, defined with its origin at the center of gravity (cg):

$$\vec{v}_{cg} = (X_b^{cg})\vec{v}_b \tag{2.45}$$

$$\vec{f}_{cg} = (X_b^{cg})^{-T}\vec{f}_b \tag{2.46}$$

The Plücker transform X_b^{cg} depends on the cg position, which can be obtained from the composite inertia I_b^c. Like all spatial vectors, $\vec{v}_{cg}$ and $\vec{f}_{cg}$ have linear and angular components:

$$\vec{v}_{cg} = \begin{bmatrix} \vec{\omega}_{cg} \\ \vec{v}_{cg} \end{bmatrix}, \quad \vec{f}_{cg} = \begin{bmatrix} \vec{\tau}_{cg} = \mathbf{0} \\ \vec{f}_{cg} \end{bmatrix} \tag{2.47}$$

We also define unit vectors in the longitudinal ($\hat{lon}$), lateral ($\hat{lat}$), and angular ($\hat{ang}$) slip directions. These are commonly aligned with the body frame x, y, and z axes but do not have to be (See Figure 2.7). Body slip velocity is parametrized over these values as follows:

$$\vec{v}_{cg,slip} = \left(p_1 \frac{f_{lon}}{f_z} V_{lon} + p_2 V_{lon}\right) \hat{lon} + \left(p_3 \frac{f_{lat}}{f_z} V_{lon}\right) \hat{lat} \tag{2.48}$$

$$\vec{\omega}_{cg,slip} = \left(p_4 \frac{f_{lat}}{f_z} V_{lon} + p_5 V_{lon} + p_6 V_{ang}\right) \hat{ang} \tag{2.49}$$

where the following are abbreviations for dot products with unit vectors:

$$f_{lon} = \vec{f}_{cg} \cdot \hat{lon} \qquad V_{lon} = \vec{v}_{cg} \cdot \hat{lon}$$
$$f_{lat} = \vec{f}_{cg} \cdot \hat{lat} \qquad V_{lat} = \vec{v}_{cg} \cdot \hat{lat}$$
$$f_z = \vec{f}_{cg} \cdot \hat{z}$$

where $\hat{z}$ is the normal force direction.

The p_1 and p_3 terms in (2.48) are direct analogues of the wheel-level relationship between slip and force in (2.41) and (2.43). The p_2 term accounts for rolling resistance, which prevents the vehicle from coasting at constant speed. Rolling resistance is proportional to normal force ([133] Section 1.2), so the f_z terms cancel out.

While only longitudinal and lateral slip are considered at the wheel level, angular slip must also be considered at the body level. The p_4 term in (2.49) accounts for oversteer/understeer behavior, which is proportional to lateral acceleration ([133] Section 5.2). The p_5 term accounts for left/right asymmetry in rolling resistance, for example due to a flat tire. The p_6 term accounts for angular slip as a result of anisotropic friction during skidding. Skid-steer vehicles cannot turn without skidding. The pseudoinverse solution assumes isotropic friction during skidding (i.e. identical friction in the longitudinal/lateral directions), but this may be false.

The parameter values ($\mathbf{p}$) depend on numerous conditions: tire pressure and stiffness, soil properties, etc. Estimating them analytically would be tedious and inaccurate, but they can be conveniently identified from experimental data (Chapter 3). In addition to slip, this enhanced model can correct for other sources of error, such as incorrect user specification of wheel radius or track width.

The spatial slip velocity at the cg must be transformed back to the body frame and added to the least-squares slip solution for $\dot{\mathbf{q}}'$, prior to integration:

$$\vec{v}_{b,slip} = (X_b^{cg})^{-1} \vec{v}_{cg,slip} \tag{2.50}$$

$$\dot{\mathbf{q}}' = \begin{bmatrix} \vec{v}_b + \vec{v}_{b,slip} \\ \dot{\boldsymbol{\theta}} \end{bmatrix} \tag{2.51}$$

Directly perturbing the body frame velocity this way could cause the wheels to penetrate or separate from the terrain surface. Alternatively, one may convert the body slip to wheel slip, then recompute the entire joint space velocity. Specifically, the x and y components of the contact point velocities ($\vec{v}_c$) are changed to nonzero values as follows, leaving the z components unchanged:

$$\mathbf{v}_c(\mathrm{x} \cup \mathrm{y}) = A_{wheel}(\mathrm{x} \cup \mathrm{y}, 1\ldots6)\vec{v}_{b,slip} \tag{2.52}$$

then $\dot{\mathbf{q}}'$ is computed using the pseudoinverse according to (2.38)

This work is not the first to predict nonzero slip in a kinematic model; for example, [46] makes wheel slip velocities linearly proportional to vehicle attitude. However, the parametrization of slip provided here is uniquely expressive and based on experimentally observed behaviors.

Limitations. This enhanced kinematic model can accurately predict the motion of real WMRs in many situations (Section 3.4.2); however, it cannot predict some extreme phenomena. For example, the enhanced kinematic model does not account for traction limits or second-

order inertial effects. Accordingly, it cannot predict vehicle motion while skidding to a stop with locked brakes.

The enhanced kinematic model also cannot predict motion during liftoff/rollover, although it does facilitate the assessment of rollover risk. According to [124] and others, if the force vector $\vec{f}_{cg}$, originating at the cg, intersects the ground plane outside of the support polygon (whose vertices are the wheel-ground contact points) then liftoff is imminent. While $\vec{f}_{cg}$ in our first-order model cannot account for all types of acceleration, it does account for centripetal acceleration and gravity, which are major contributors to sideways rollover.

If necessary, extreme phenomena *can* be predicted accurately using our dynamic model formulation (Section 2.5.3); however, under normal operating conditions an enhanced kinematic model may provide comparable accuracy at less computational cost. In addition, an enhanced kinematic model has fewer parameters (i.e. just six in $\mathbf{p}$) than a comparably expressive dynamic model, making it simpler/faster to calibrate.

2.4 Dynamic Model Formulation

This section explains our dynamic model formulation. First, Section 2.4.1 presents an unconstrained WMR dynamics formulation and the recursive algorithms used to compute the joint space inertia and bias force terms. Section 2.4.2 presents our alternative, constrained dynamics formulation. Section 2.4.3 explains our force-balance optimization technique, which resolves overconstraint and nonlinearity issues. Finally, Section 2.4.4 discusses the overall simulation procedure and identifies techniques for faster computation.

2.4.1 Unconstrained Dynamics

In related work, most model WMR dynamics with an unconstrained ordinary differential equation (ODE) of the form:

$$M(\mathbf{q})\ddot{\mathbf{q}}' + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}') = \boldsymbol{\tau} \tag{2.53}$$

$\dot{\mathbf{q}}'$ and $\ddot{\mathbf{q}}'$ in (2.53) are the joint space velocity and acceleration vectors, explained in Section 2.2.1. $\boldsymbol{\tau}$ is a vector of applied forces/torques. M is the joint space inertia matrix; it can be computed using Algorithm 3, a modified version of the Composite Rigid Body Algorithm (CRBA). Featherstone [36] originally developed the CRBA for robotic manipulators; it is an efficient, recursive algorithm for linked rigid bodies. $\mathbf{c}$ is the joint space bias force, which accounts for velocity-dependent Coriolis and centripetal terms. $\mathbf{c}$ can be computed using Algorithm 4, a modified version of the Recursive Newton Euler Algorithm (RNEA). Both the CRBA and RNEA use spatial vector algebra (SVA) for concise mathematical expressions.

Featherstone and Orin [37] summarize the intuition behind the original CRBA and RNEA. To our knowledge, no one has previously applied these algorithms to WMRs, though McMillan and Orin [86] used them to simulate a multilegged vehicle. Our versions are modified from the canonical versions in [37] to account for a mobile base, only single DOF joints, and the massless contact frames.

Algorithm 3 Composite Rigid Body Algorithm (CRBA): joint space inertia

1: **for** $i = 1$ to n_f **do** $I_i^c \leftarrow I_i$
2: **for** $i = n_f$ to 2 by -1 **do**
3: $\quad I_{p(i)}^c \leftarrow I_{p(i)}^c + (X_{p(i)}^i)^T I_i^c (X_{p(i)}^i)$
4: **end for**
5: $M \leftarrow [0]$
6: $M(1...6, 1...6) \leftarrow I_1^c$
7: **for** $i = 2$ to n_f **do**
8: $\quad \vec{f}^c \leftarrow I_i^c(*, s(i))$
9: $\quad M(i+5, i+5) \leftarrow \vec{f}^c(s(i))$
10: $\quad j \leftarrow i$
11: $\quad$ **while** $j > 1$ **do**
12: $\quad\quad \vec{f}^c \leftarrow (X_{p(i)}^i)^T \vec{f}^c$
13: $\quad\quad j \leftarrow p(j)$
14: $\quad\quad$ **if** $j = 1$ **then**
15: $\quad\quad\quad M(1...6, i+5) \leftarrow \vec{f}^c$
16: $\quad\quad\quad M(i+5, 1...6) \leftarrow M(i+5, 1...6)^T$
17: $\quad\quad$ **else**
18: $\quad\quad\quad M(j+5, i+5) \leftarrow \vec{f}^c(s(j))$
19: $\quad\quad\quad M(i+5, j+5) \leftarrow M(j+5, i+5)$
20: $\quad\quad$ **end if**
21: $\quad$ **end while**
22: **end for**

To compute M in Algorithm 3, we first recursively compute the *composite* spatial inertia of the subtree rooted at each frame (line 3). Conceptually, I_i^c is the inertia of frame i if frame i and the subtree rooted to it were fused into a single rigid body. The Plücker transforms ($X_{p(i)}^i$) are obtained by converting the homogeneous transforms ($T_i^{p(i)}$) computed in Algorithm 1. The upper left 6×6 block of M is the composite inertia of the body frame (line 6). The remaining columns equal the generalized force required to accelerate each joint individually by one unit, in a system where all velocities (and acceleration-independent forces) are zero. $s(i)$ maps the joint type of frame i to a spatial vector index (RX=1, RY=2, RZ=3, PX=4, PY=5, PZ=6). Rows are set equal to columns (lines 16 and 19) because M is symmetric.

To compute $\mathbf{c}$ in Algorithm 4, we first recursively compute the spatial velocity and acceleration of each frame in a system where $\ddot{\mathbf{q}}' = 0$ (lines 7 and 8). Coriolis and centripetal forces are computed for each frame (line 10), and recursively summed to obtain the joint space bias force (line 14). n_t is the total number of frames, including the n_f user-defined frames and the n_w contact frames (one per wheel). Contact frames are appended to the list of user-defined frames; their indices are $\{n_f + 1, \ldots, n_f + n_w\}$ and their parent indices $p(i)$ are the wheel frame indices. Note that external forces can be applied to any frame on line 10. Gravitational forces could be applied here, but it is computationally cheaper to instead set the base acceleration to the negative of gravitational acceleration (line 2).

Algorithm 4 Recursive Newton Euler Algorithm (RNEA): joint space bias force

1: $\vec{v}_1 \leftarrow \dot{\mathbf{q}}'(1\ldots6)$
2: $\vec{a}_1 \leftarrow -^b\vec{g}$
3: **for** $i = 1$ to n_t **do**
4: **if** $i \geq 1$ **then**
5: $\vec{h} \leftarrow \mathbf{0}$
6: **if** $i \leq n_f$ **then** $\vec{h}(s(i)) \leftarrow \dot{\mathbf{q}}'(i+5)$
7: $\vec{v}_i \leftarrow X^i_{p(i)}\vec{v}_{p(i)} + \vec{h}$
8: $\vec{a}_i \leftarrow X^i_{p(i)}\vec{a}_{p(i)} + \vec{v}_i \times_m \vec{h}$
9: **end if**
10: $\vec{f}_i \leftarrow I_i\vec{a}_i + \vec{v}_i \times_f I_i\vec{v}_i(+\vec{f}^{ext}_i)$
11: **end for**
12: **for** $i = n_t$ to 2 by -1 **do**
13: **if** $i \leq n_f$ **then** $\mathbf{c}(i+5) \leftarrow \vec{f}_i(s(i))$
14: $\vec{f}_{p(i)} \leftarrow \vec{f}_{p(i)} + (X^i_{p(i)})^T\vec{f}_i$
15: **end for**
16: $\mathbf{c}(1\ldots6) = \vec{f}_1$

Note that if a frame's index in the frame list is i, its corresponding index in a joint space vector is $i + 5$ (e.g. line 6 in Algorithm 4). That is because the body frame "joint" has 6-DOF and thus six indices in joint space, but only one index in the frame list.

One can simulate WMR dynamics using only the ordinary differential equation in (2.53). Wheel-ground contact forces can be computed using any model based on ($\mathbf{q}$, $\dot{\mathbf{q}}'$) values at the current step, then added as external forces to the contact frames inside the RNEA. Actuator forces can be included in the $\boldsymbol{\tau}$ vector. Unfortunately, the dynamics will be very stiff, such that an adaptive integrator and very small time steps are required to maintain stability [7][118]. The system is more stable if contact and actuator forces are handled via constraints.

2.4.2 Constrained Dynamics

This section presents our alternative differential algebraic equation (DAE) formulation of WMR dynamics.

Velocity level constraints. This alternative formulation enforces velocity kinematic constraints of the form:

$$A(\mathbf{q})\dot{\mathbf{q}}' = \mathbf{v} \tag{2.54}$$

(2.54) comprises three constraint types: wheel-ground contact, actuator, and holonomic joint constraints:

$$\begin{bmatrix} A_{wheel} \\ A_{act} \\ A_{joint} \end{bmatrix} \dot{\mathbf{q}}' = \begin{bmatrix} \mathbf{v}_c \\ \mathbf{v}_{act} \\ \mathbf{v}_{joint} \end{bmatrix} \tag{2.55}$$

The wheel-ground contact and joint constraints used in the dynamic model formulation are

identical to those used in the kinematic formulation (Sections 2.2.2 and 2.2.3). The additional actuator constraints are relatively simple. The user specifies which joints are actuated in the frame information, and those joint rates are constrained to equal $\mathbf{v}_{act}$. The Jacobian A_{act} for a single joint is simply a row vector of zeros, except for a 1 at the index of that joint's rate in $\dot{\mathbf{q}}'$. For example, for the Rocky 7 front right wheel (A2):

$$A_{act} = [[0\ 0\ 0\ 0\ 0\ 0]\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0] \tag{2.56}$$

The index of A2 is 10 in the frame list (Table 2.2) and thus 15 in $\dot{\mathbf{q}}'$.

The challenge to specifying the constraint equation (2.55) in the dynamic model is correctly setting values in $\mathbf{v}$ on the right hand side. For the wheel-ground contact constraints, setting the x,y elements of $\vec{v}_c$ for each wheel to zero prevents slip, but we desire a dynamic model that realistically predicts nonzero slip. Setting the z element of $\vec{v}_c$ to zero could prevent penetration of the wheel into the terrain, but it could also prevent the wheel from lifting off. We desire a dynamic model that correctly predicts liftoff and rollover.

Setting $\mathbf{v}_{act}$ for the actuator constraints is also challenging. We could set $\mathbf{v}_{act}$ to commanded rates, but real actuator/powertrain systems cannot follow arbitrary commands. Due to hardware limitations on power or torque, there is a transient response when commands change, and some commanded rates may be unreachable.

Finally, we could set $\mathbf{v}_{joint}$ such that joint constraints are rigidly enforced, but we may want to account for spring/damper effects in the real system.

Section 2.4.3 presents our solution to this problem; we choose values for $\mathbf{v}$ that satisfy user-specified models relating these velocities to constraint forces.

Acceleration level constraints. Recall that (2.55) comprises both nonholonomic and holonomic constraints, but all are expressed in identical form: as a linear function of the joint space velocity. The nonholonomic constraints (on wheel slip, actuated joint rates) were already of this form. The holonomic constraints (on terrain contact, joint displacements) were effectively converted into this form by differentiation with respect to time.

This technique for converting holonomic constraints to nonholonomic form was presented by Yun and Sarkar [138]:

$$c(\mathbf{q}) = 0 \tag{2.57}$$

$$\frac{dc(\mathbf{q})}{dt} = \frac{\partial c}{\partial \mathbf{q}}\dot{\mathbf{q}} = 0 \tag{2.58}$$

The equation (2.58) only enforces the holonomic constraint to first order and requires stabilization to prevent drift.

To enforce the velocity-level constraints (2.54) in a second-order, dynamic simulation they must be converted to constraints on acceleration. We can do this in discrete time as follows. Here, numbers in brackets denote the time step ($[i]$ for the current step, $[i+1]$ for the next step).

$$A\dot{\mathbf{q}}'[i+1] = \mathbf{v}[i+1] \tag{2.59}$$

$$A(\dot{\mathbf{q}}'[i] + \ddot{\mathbf{q}}'[i]\Delta t) = \mathbf{v}[i+1] \tag{2.60}$$

$$A\ddot{\mathbf{q}}'[i] = \frac{\mathbf{v}[i+1] - A\dot{\mathbf{q}}'[i]}{\Delta t} = \mathbf{b} \tag{2.61}$$

Essentially, we require that the constraint on $\dot{\mathbf{q}}'$ be satisfied at the *next* time step. This matches the impulse-momentum time stepping technique used by Stewart and Trinkle [123] and Open Dynamics Engine. (2.61) makes an approximation by holding A constant over the step, i.e. it assumes that $A(\mathbf{q}[i+1]) \approx A(\mathbf{q}[i])$.

A continuous time alternative is to differentiate the velocity-level constraint equation with respect to time:

$$\frac{dA(\mathbf{q})\dot{\mathbf{q}}'}{dt} = \mathbf{0} \tag{2.62}$$

$$A\ddot{\mathbf{q}}' + \dot{A}\dot{\mathbf{q}}' = \mathbf{0} \tag{2.63}$$

$$A\ddot{\mathbf{q}}' = -\dot{A}\dot{\mathbf{q}}' \tag{2.64}$$

In effect, this requires differentiating nonholonomic constraints once and holonomic constraints twice. Unlike (2.61), just enforcing (2.64) could allow drift in the nonholonomic constraints [138].

We combine the acceleration level constraints (2.61) with the ordinary differential equation (2.53) to obtain the constrained WMR dynamics in DAE form:

$$\begin{bmatrix} M & A^T \\ A & [0] \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}' \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau} - \mathbf{c} \\ \mathbf{b} \end{bmatrix} \tag{2.65}$$

$\boldsymbol{\lambda}$ in equation (2.65) is a vector of Lagrange multipliers, or constraint forces. Wheel-ground contact and actuator forces are now included implicitly through $\boldsymbol{\lambda}$, rather than being explicitly added inside the RNEA or through $\boldsymbol{\tau}$.

In contrast to (2.65), some in related work compute the nullspace of A to derive constraint-free dynamics [105][138]. This is not possible in our formulation due to overconstraint and nonlinearity issues discussed next.

2.4.3 Force-Balance Optimization

There are two primary issues with the constrained dynamics formulation expressed in (2.65): *overconstraint* and *nonlinearity* of the user-specified models relating constraint forces to velocities. These models must be satisfied when setting the values $\mathbf{v}$ in (2.54), of which $\mathbf{b}$ in (2.65) is a function. This section presents a novel "force-balance optimization" technique to resolve these issues, which is perhaps the most important theoretical contribution in our dynamic model formulation.

Throughout this section, we refer to sets of row indices in the constraint equation. For convenience these sets are listed in Table 2.3.

Overconstraint and rank deficiency. Most WMRs are overconstrained, meaning that the matrix A in (2.65) has more rows than columns, such that not all constraints can be satisfied. Many address overconstraint in an ad hoc way. In 2D, reduced models are often substituted for overconstrained ones; for example, a four-wheeled Ackerman steered car is modeled as a bicycle ([133] Section 5.5), or a skid-steer vehicle (with four or more wheels) is modeled as a two-wheel differential drive vehicle [82]. When Choi and Sreenivasan [26] and Chakraborty

Table 2.3: Sets of row indices in the constraint equation

Set name	Description
wheel	wheel-ground contact constraints
act	actuator constraints
joint	holonomic joint constraints
indep	linearly independent subset of constraints
input	constraints with input variable in force-balance optimization (input $\subseteq$ indep)

and Ghosal [23] performed constrained 3D kinematic simulations, they carefully designed their vehicles to avoid overconstraint by adding degrees of freedom (variable length axle, passive variable camber).

When A has more rows than columns, some rows must be linearly dependent. However, even in the absence of overconstraint, some rows may still be linearly dependent, making A rank deficient. This occurs when wheel-ground contact frames align, such that they produce redundant constraints.

Ideally, overconstraint and rank deficiency should be handled in a general way that accommodates any vehicle configuration. They should also be handled in a dynamic way, because the size of A will change as wheels break/reestablish contact with the terrain, and the rank will change with wheel/contact frame orientation.

The first step of our solution is to extract a linearly independent subset of constraints (or rows of A) by QR decomposition. To obtain a well-conditioned subset, a QR algorithm with pivoting should be used:

$$QR = A^T E \tag{2.66}$$

$$\mathbf{e} = [1 \ \ldots \ n_c]E \tag{2.67}$$

$$\text{indep} = \mathbf{e}(\{i : |R(i,i)| > \epsilon\}) \tag{2.68}$$

Pivoting is responsible for the permutation matrix E. The vector $\mathbf{e}$ also encodes the permutation information; it is a reordering of the indices from 1 to n_c (the number of constraints). The set indep contains the indices of constraints in the linearly independent subset. Only this independent subset of constraints is enforced, such that (2.65) becomes:

$$\begin{bmatrix} M & A(\text{indep}, *)^T \\ A(\text{indep}, *) & [0] \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}' \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau} - \mathbf{c} \\ \mathbf{b}(\text{indep}) \end{bmatrix} \tag{2.69}$$

The size of $\boldsymbol{\lambda}$ is now the number of constraints in the independent subset.

By itself, this fix would cause undesirable behavior; the eliminated constraints would allow some wheels to slip and deviate from the terrain surface uncontrollably. However, the force-balance optimization technique discussed below ensures that *all* constraints are enforced according to user-specified models.

Nonlinearity. We allow users to specify three types of models: wheel-ground contact, actuator, and joint constraint. These models relate $\mathbf{v}$ in (2.54) to constraint forces. However, realistic models are nonlinear, making them difficult to integrate with our linearized dynamics equation (2.69).

(Wheel-ground contact models). Open Dynamics Engine supports a simple contact model that allows nonzero slip prediction in a limited way. Through a "force-dependent slip" parameter, users may specify some slip that is *linearly* proportional to the tangential contact force. OpenDE accomplishes this through "constraint force mixing," which effectively replaces the block of zeros in (2.65) with a diagonal matrix, introducing $\boldsymbol{\lambda}$ into the constraint equation.

OpenDE also supports an approximation to Coulomb's law of friction. Specifically, OpenDE supports a "pyramid" approximation to a Coulomb friction cone, which limits tangential force in two orthogonal directions:

$$\begin{aligned} f_x &\leq \mu_x f_z \\ f_y &\leq \mu_y f_z \end{aligned} \tag{2.70}$$

where μ_x and μ_y are coefficients of friction, and f_z is the normal force. A true Coulomb friction limit is of the form: $\left\| [f_x \ f_y]^T \right\| \leq \mu f_z$

While this simple model is sufficient for many applications, more realistic wheel-ground contact modeling is needed to accurately predict WMR motion. Most wheel-ground contact models in the literature are nonlinear functions relating force to wheel slip, tire compression/sinkage, and other parameters. Our solution can enforce any contact model, if expressed as a function of the form:

$$\vec{f}_c = \vec{f}(\vec{v}_c, R\omega, \Delta z) \tag{2.71}$$

In (2.71), $\vec{v}_c$ is the contact point velocity and $\vec{f}_c$ is the contact force (exerted on the wheel by the ground), both expressed in local coordinates. $R\omega$ is the product of wheel radius and angular velocity, which is required to compute slip ratio and slip angle. Δz is the contact height error defined in Section 2.2.1. A negative Δz can indicate tire compression, or sinkage into the terrain.

Robotics literature contains many terramechanics-based models for rigid wheels in loose soil [52][31][56]; most are based on early work by Bekker [11] and Wong and Reece [134]. Automotive literature contains many empirical models for pneumatic tires [19]; the "magic formula" developed by Pacejka and Bakker [94] is widely used. All models we have encountered in the literature can be expressed in the form of (2.71). Note that the function cannot typically be inverted; i.e. a contact force may map to multiple slip velocities. More information about the wheel-ground contact models used in our tests is provided in Appendix C.

(Actuator models). Open Dynamics Engine supports "motors" for revolute and prismatic joints; these bring joint rates up to a desired value in a single time step using constraints. Through the "dParamFMax" parameter, OpenDE also supports a fixed limit on motor force/torque.

$$|f_{act}| \leq f_{max} \tag{2.72}$$

Realistic models of the actuators that power WMRs are more complex and nonlinear.

Just as our solution allows great flexibility in the choice of wheel-ground contact model, it also supports any actuator model, if expressed as a function of the form:

$$\mathbf{f}_{act} = \mathbf{f}(\boldsymbol{\theta}_{act}, \dot{\boldsymbol{\theta}}_{act}, \mathbf{u}, \ldots) \tag{2.73}$$

In (2.73), $\dot{\boldsymbol{\theta}}$ is the vector of actuated joint rates, and $\mathbf{f}_{act}$ is the vector of forces/torques exerted by the actuators. $\mathbf{u}$ is the vector of commanded joint rates. Because this is a vector-valued

function for *all* actuated joints, coupling between joints is possible (e.g. for synchronous drive robots).

The function in (2.73) can encode, for example, the torque-speed characteristics of an electric motor or of an internal combustion engine and torque converter (see e.g. [133] Section 3.3). The model also encloses the joint-level controller; this could be a proportional-integral-derivative (PID) controller, for example, with additional inputs for prior errors.

(Joint constraint models). We support any model expressed as a function of the form:

$$\mathbf{f}_{joint} = \mathbf{f}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \tag{2.74}$$

$\mathbf{f}_{joint}$ is the vector of constraint forces. These forces are often linearly dependent on the input joint displacements and rates (e.g. for a spring/damper suspension joint); however, more complex models including hysteresis and other nonlinear effects are possible.

Force-balance optimization. Through force-balance optimization, we solve for $\mathbf{v}$ such that the user-specified models relating these velocities to the constraint forces ($\boldsymbol{\lambda}$) are satisfied.

Let $\mathbf{x}$ denote the subvector of $\mathbf{v}$ to be solved for, or the inputs to the force-balance optimization:

$$\mathbf{x} = \mathbf{v}(\text{input}) \tag{2.75}$$

For now, assume that the input set is identical to the indep set, which specifies the linearly independent subset of constraints. Later in Section 2.4.4 we explain how some inputs may be omitted when assuming ideal actuators.

We solve for the input vector ($\mathbf{x}^*$) that minimizes the following "force-balance" objective function:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \; L(\mathbf{x}) \tag{2.76}$$

$$L(\mathbf{x}) = \|\mathbf{e}(\mathbf{x})\|^2 = \mathbf{e}(\mathbf{x})^T\mathbf{e}(\mathbf{x}) \tag{2.77}$$

$$\mathbf{e}(\mathbf{x}) = A(\text{indep}, *)^T\boldsymbol{\lambda} - A^T\mathbf{f}_{model} \tag{2.78}$$

$$\mathbf{f}_{model} = \begin{bmatrix} \mathbf{f}_c \\ \mathbf{f}_{act} \\ \mathbf{f}_{joint} \end{bmatrix}$$

According to (2.77), we minimize the error vector $\mathbf{e}$ in a least-squares sense. According to (2.78), error is defined as the difference between constraint forces ($\boldsymbol{\lambda}$) and model forces ($\mathbf{f}_{model}$), i.e. those output by the user-specified models in (2.71), (2.73), and (2.74). The space of model forces will typically be larger than the space of constraint forces (due to the elimination of linearly dependent constraints), but both are projected onto joint space for comparison.

Both the constraint and model forces are dependent on $\mathbf{x}$. (2.69) can be solved for the constraint forces ($\boldsymbol{\lambda}$) as follows:

$$\boldsymbol{\lambda} = C^{-1}\mathbf{d} \tag{2.79}$$

$$C = A(\text{indep}, *)M^{-1}A(\text{indep}, *)^T \tag{2.80}$$

$$\mathbf{d} = \mathbf{b}(\text{indep}) - A(\text{indep}, *)M^{-1}(\boldsymbol{\tau} - \mathbf{c}) \tag{2.81}$$

Recall that $\mathbf{b}$ in (2.81) is a function of $\mathbf{v}$ per (2.61), and thus a function of the input vector $\mathbf{x}$.

In accordance with (2.61), model forces ($\mathbf{f}_{model}$) are also computed based on the value of $\dot{\mathbf{q}}'$ at the *next* time step ($i+1$) vs. the current time step (i). After solving for $\boldsymbol{\lambda}$ per (2.79), we solve for the joint space acceleration ($\ddot{\mathbf{q}}'$) as follows:

$$\ddot{\mathbf{q}}' = M^{-1}(\boldsymbol{\tau} - \mathbf{c} - A(\text{indep}, *)^T \boldsymbol{\lambda}) \tag{2.82}$$

and then update joint space velocity:

$$\dot{\mathbf{q}}'[i+1] = \dot{\mathbf{q}}'[i] + \ddot{\mathbf{q}}'\Delta t \tag{2.83}$$

Inputs for the wheel-ground contact, actuator, and joint constraint models are then obtained from $\dot{\mathbf{q}}'[i+1]$. For the wheel-ground contact model:

- contact point velocities ($\vec{v}_c$) for all wheels are obtained from $\mathbf{v}_c = A(\text{wheel}, *)\dot{\mathbf{q}}'[i+1]$.
- wheel angular velocities (ω) are obtained directly from $\dot{\mathbf{q}}'[i+1]$.
- updated contact height errors are computed as follows: $\Delta z[i+1] = \Delta z[i] + v_z \Delta t$, where v_z is the third element of $\vec{v}_c$.

For the actuator model and joint constraint models:

- joint rates ($\dot{\boldsymbol{\theta}}$) are obtained directly from $\dot{\mathbf{q}}'[i+1]$.
- updated joint displacements are computed as follows: $\boldsymbol{\theta}[i+1] = \boldsymbol{\theta}[i] + \dot{\boldsymbol{\theta}}\Delta t$.

Minimization of the objective function $L(\mathbf{x})$ in (2.77) to zero indicates that all user-specified models are satisfied. This minimization can be done efficiently and robustly using Newton's method combined with a line search algorithm. The input vector is updated according to:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha[HL(\mathbf{x}_k)]^{-1}\nabla L(\mathbf{x}_k) \tag{2.84}$$

where $\nabla L(\mathbf{x}_k)$ is the gradient of the objective function, and $HL(\mathbf{x}_k)$ is the Hessian. Computing these requires that the user-specified functions be differentiable with respect to their inputs.

The step size α in (2.84) is bounded between 0 and 1. We use a line search algorithm to find an α such that the strong Wolfe conditions are satisfied, as explained in Appendix D. Other optimization methods are possible such as the Levenberg-Marquadt algorithm.

Finally we note that, unlike Open Dynamics Engine, we do not solve for constraint forces ($\boldsymbol{\lambda}$) via a Linear Complementarity Problem (LCP). OpenDE solves for contact forces in two-steps. First, normal forces are computed *assuming frictionless contacts*. An LCP is solved with inequality constraints that prohibit negative normal force, and complementarity conditions that normal force be zero if separation occurs. These normal forces are then used to set limits for tangential friction forces per (2.70). Then, the LCP is solved again with tangential forces bounded *independently* in two orthogonal directions. (OpenDE User Guide [116] Section 3.11.1)

Some have attempted to enforce more sophisticated contact models in OpenDE by modulating the coefficient of friction [42], but such attempts are limited by OpenDE's inherently independent treatment of force in each direction in its LCP formulation. In our method, forces in all directions are solved for simultaneously, such that more complex dependencies are permitted (Appendix C). These are specified by the wheel-ground contact model and satisfied via force-balance optimization.

2.4.4 Dynamic Simulation Process

This section provides an overview of the dynamic simulation process. Techniques for faster computation and limitations of the dynamic model are also identified.

The dynamic simulation process is summarized by the flowchart in Figure 2.6. In addition to an initial state ($\mathbf{q}$), an initial joint space velocity ($\dot{\mathbf{q}}'$) is also required. The "forward dynamics" block is the calculation of joint space acceleration via the constrained dynamics and force-balance optimization, as explained in Section 2.4.3.

We use symplectic Euler integration to update the state:

$$\dot{\mathbf{q}}'[i+1] = \dot{\mathbf{q}}'[i] + \ddot{\mathbf{q}}'[i]\Delta t \tag{2.85}$$

$$\mathbf{q}[i+1] = \mathbf{q}[i] + V(\mathbf{q}[i])\dot{\mathbf{q}}'[i+1]\Delta t \tag{2.86}$$

where the V matrix is defined in equation (2.14). The difference between symplectic and conventional Euler integration is that the velocity at step $[i+1]$ vs. $[i]$ is used to update state. Symplectic (a.k.a. semi-implicit) Euler integration is efficient and effective at conserving system energy [43]. Other integration methods are possible, but note that symplectic Euler is assumed internally during force-balance optimization (i.e. in the calculation of some inputs for the user-specified models).

Techniques for faster computation. Faster simulation will always be advantageous, even as computing hardware improves. In a planning context, faster simulation enables a longer planning horizon, or the evaluation of more candidate trajectories. We have identified many techniques to speed up simulation without impacting fidelity.

The most expensive calculations are of the cost $L(\mathbf{x})$ and gradient $\nabla L(\mathbf{x})$ for force-balance optimization, due to the many matrix-matrix and matrix-vector multiplications required. Time can be saved by caching intermediate variables for use in subsequent iterations of Newton's method. For example, the constraint forces $\boldsymbol{\lambda}$ must be recomputed every time the cost function is called, but C in (2.80) and its decomposition need only be computed once per time step. Likewise, the constant part of $\mathbf{d}$ in (2.81) (i.e. the part not dependent on $\mathbf{x}$) need only be computed once:

$$\mathbf{d} = \mathbf{b}(\text{indep}) + \mathbf{d}_{const} \tag{2.87}$$

$$\mathbf{d}_{const} = -A(\text{indep}, *)M^{-1}(\boldsymbol{\tau} - \mathbf{c}) \tag{2.88}$$

Because the matrices M, C, and $HL(\mathbf{x})$ are all symmetric positive definite, their Cholesky decompositions can be computed relatively cheaply. Their inverses need not be computed explicitly; equations requiring their inverses can be solved using their Cholesky decompositions and forward/back substitution.

We exploit the continuity of WMR motion as a function of time, i.e. that velocity changes little between time steps as step size approaches zero. Specifically, we initialize the inputs $\mathbf{x}$ in the force-balance optimization to their values at the previous time step. This reduces the number of Newton's method iterations required. When operating at or near steady-state, this initial guess is sometimes close enough such that minimization is not required at all.

Wheel-ground contact models can be arbitrarily complex. Terramechanics-based models in particular require the integration of stress distributions along the wheel surface. Fortunately, a

lookup table can often be precomputed. According to the function's required form in (2.71), the inputs and thus the lookup table are up to 5D; however, a lower dimensional table can often be obtained by parametrizing over slip ratio and angle. See Appendix C for details.

Additional speed up is possible with minor approximations in the dynamics. First, like Open Dynamics Engine we can assume *ideal actuators*. Ideal actuators are capable of supplying any force/torque necessary to meet commanded joint rates at every time step. When assuming ideal actuators, we automatically include all actuator constraints in the linearly independent subset. Accordingly we can omit the rows and columns of A corresponding to actuated joints when computing the QR decomposition in (2.66). We can also omit the $\mathbf{v}_{act}$ inputs from $\mathbf{x}$ in the force-balance optimization, and set them directly equal to the commanded joint rates ($\mathbf{u}$). Finally, we set the elements of $\mathbf{e}$ in (2.78) to zero, such that actuator force/torque has no effect on cost. When the WMR is operating within the capabilities of its actuators, assuming ideal actuators has minimal impact on fidelity; however, the elimination of inputs can significantly speed up the force-balance optimization.

Second, for vehicles with no articulated suspension the joint space inertia matrix M may never change. For other vehicles, changes in M may be negligible over the prediction horizon. In these cases M and its inverse (or decomposition) can be computed once and reused on subsequent time steps. Additionally, vehicle mass is usually concentrated in the body frame. As a result, joint space bias forces may be negligible for internal articulations. If so, instead of using the RNEA (Algorithm 4), the joint space bias force $\mathbf{c}$ can be approximated as follows:

$$\mathbf{c} = \begin{bmatrix} -I_b^c \vec{g} + \vec{v}_b \times_f I_b^c \vec{v}_b \\ \mathbf{0} \end{bmatrix} \tag{2.89}$$

Where I_b^c is the composite inertia of the WMR rooted at the body frame, $\vec{g}$ is the spatial acceleration of gravity, and $\vec{v}_b$ is the spatial velocity of the body frame (i.e. the first six elements of $\dot{\mathbf{q}}'$). All are expressed in body coordinates. This approximation considers the WMR to be a single, fused rigid body when computing the bias force. Note, however, that the joints are still free to articulate due to other forces.

Finally, for a *non-iterative* dynamic simulation, we can take a single Newton's method step per time step in the force-balance optimization (i.e. perform (2.84) just once). As a result, the cost function in (2.78) will not always be minimized to zero, and so the wheel-ground contact, actuator, and joint constraint models will not be perfectly satisfied at every time step. However, error will be continually driven to zero and reach zero under steady-state conditions. This is analogous to our use of Baumgarte's method to stabilize contact height error in the kinematics formulation, except that here we stabilize force-balance error.

Limitations. Depending on the choice of wheel-ground contact, actuator, and joint constraint models the force-balance optimization problem may be non-convex. In that case there is a risk of converging to local minima at which $L(\mathbf{x}) > 0$ and the models are not satisfied. In practice we encounter this only rarely. For example, we have observed it occasionally for a highly-articulated platform on rough terrain, at time steps when wheels reestablish contact with the ground. In these instances convergence to zero can be achieved by temporarily decreasing the step size. Alternatively, one could revert to convex models.

2.5 Evaluation of Model Formulations

This section assesses the functionality, stability, and computation speed of our WMR model formulations vs. state-of-the-art alternatives in simulation. Model accuracy (after calibration) is assessed in physical experiments in Section 3.4.

First, Sections 2.5.1 and 2.5.2 compare our DAE kinematics and dynamics formulations to conventional unconstrained ones. Next, Section 2.5.3 demonstrates the prediction of extreme phenomena like loss of traction and rollover using our dynamics formulation. Finally, Section 2.5.4 benchmarks the computation speed of our models vs. Open Dynamics Engine models.

We have implemented our model formulations in both MATLAB and C++ software libraries [109]. We found that careful implementation in C++ is significantly faster (Section 2.5.4).

2.5.1 Comparison to Unconstrained Kinematics

Here we compare our "stabilized DAE" method of kinematic motion prediction to the traditional, unconstrained method used by [125][46] and others. While some have used DAEs for the kinematic simulation of particular WMRs [26][23][34][39], our work is the first to combine DAEs with a general method for kinematics derivation. The term "stabilized" refers to our use of Baumgarte's stabilization method to prevent drift in the holonomic constraints (Section 2.2.5).

In the traditional method, each simulation step comprises an unconstrained motion followed by "terrain fitting," in which elevation, attitude, and passive joint displacements are solved for by minimization of a contact error objective function (Sections 2.1.1, 2.2.4). This nonlinear minimization can be computationally expensive, and it can induce unwanted wheel slip.

Here we compare these two methods in a simulation of the Rocky 7 rover traversing rough terrain. Tarokh and McDermott [125] also use the Rocky 7 rover to demonstrate their approach. Frame information for Rocky 7 is provided in Section 2.2.1. The rover's rocker-bogie design allows all six wheels to stay in contact on uneven terrain.

In this test, we commanded the rover to drive straight (open-loop) at 0.5 m/s for 10 seconds. We implemented a naïve controller that assumed flat ground and drove all wheels at equal speeds. The rover actually traversed a randomly-generated rough terrain surface with a root mean square (RMS) height of .05 m and a correlation length of 0.2 m.

Quantitative results are presented in Table 2.4. These are averaged over ten trials, each with a unique rough terrain surface (i.e. generated with a different random seed). We conducted the test using both simulation methods and step sizes ranging from .01 to .20 seconds. Trials using the unconstrained method and .01 and .02 second step sizes were omitted due to excessive computational cost.

The *stabilized DAE* simulation was performed as described in Section 2.2.5. The rover would have driven straight on flat ground; however, on rough terrain, enforcement of the wheel-ground contact constraints caused deviation off course (Figure 2.8).

The *unconstrained* method was performed as described in [125]. At each step, position was updated by integrating the nominal velocity of the body frame (in body coordinates). Then, the terrain fitting minimization was performed using MATLAB's built-in *fminsearch* function. This method *incorrectly* predicted that the rover drove straight as commanded.

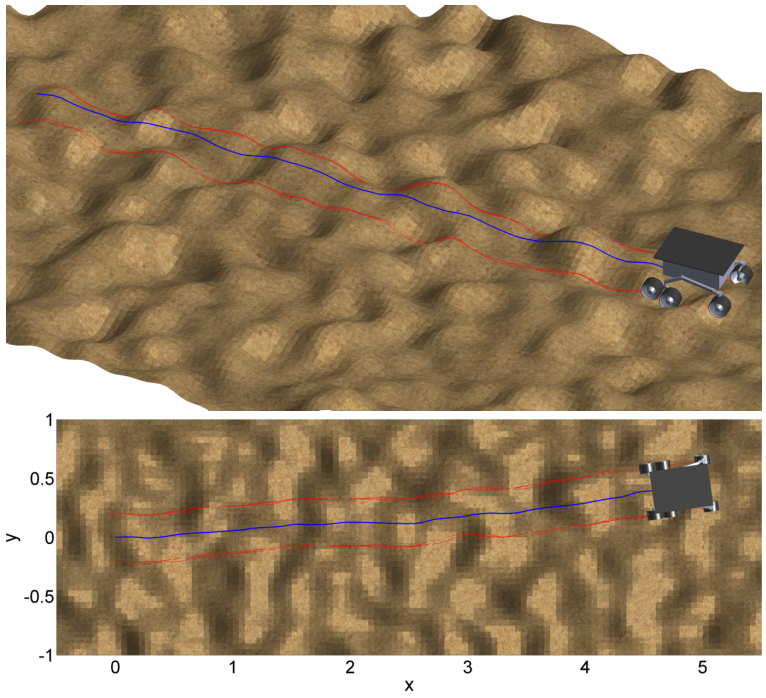


Figure 2.8: Animation screenshots for a kinematic simulation of the Rocky 7 rover traversing rough terrain. Blue lines trace body frame position, red lines trace wheel-ground contact point positions. The rover is commanded to drive straight, but the stabilized DAE method correctly predicts deviation.

Table 2.4: Rocky 7 kinematic simulation errors and computation times

method	step size (s)	mean $\|\vec{v}_c\|$ (m/s)[1]	pos. err. (m)[2]	yaw err. (rad)[3]	motion time[4]	total time[5]
stabilized DAE	0.01	0.0703	-	-	0.0438	0.1655
	0.02	0.0708	0.0136	0.0073	0.0217	0.0823
	0.04	0.0721	0.0412	0.0268	0.0109	0.0413
	0.10	0.0830	0.1286	0.0703	0.0044	0.0166
	0.20	0.1650	0.3004	0.1554	0.0023	0.0085
unconstrained	0.04	0.1158	0.5199	0.1367	1.9639	1.9952
	0.10	0.1188	0.5199	0.1367	1.3569	1.3692
	0.20	0.1246	0.5200	0.1367	1.0541	1.0604

1. mean contact point velocity (ideally zero)
2. Euclidean distance error of final position (relative to stabilized DAE simulation with .01 s step size)
3. absolute error of final yaw
4. computation time required for motion prediction only
5. computation time required for total simulation (including collision detection)

All results are averaged over ten trials. Computation times are for MATLAB implementation, and are normalized such that < 1.0 is faster than real-time.

The "mean $\|\vec{v}_c\|$" column in Table 2.4 is the mean magnitude of contact point velocity over all wheels and all time steps. Ideally in a no-slip kinematic simulation this value would be zero, but some nonzero velocity is unavoidable due to overconstraint and drift stabilization. For a .04 s step size, the mean contact point velocity using our DAE method is nearly 40% less than using the unconstrained method. For both methods, contact point velocity increases with step size due to linearization error.

Failure to constrain contact point velocities results in large motion prediction errors over time. After 10 seconds, mean errors in the predicted position and yaw of the rover grow to .52 m and .1367 radians (7.8°) when using the unconstrained method. Here, error is measured relative to predicted motion using the stabilized DAE method and the .01 s step size; the DAE method is objectively more correct than the unconstrained method because it controls wheel slip, and linearization errors are least for the smallest step size. Figure 2.9 plots the error results. Using our DAE method, errors grow with step size but are smaller than if using the unconstrained method for most step sizes; e.g. for a .04 s step size position and yaw errors are 92% and 80% smaller.

The last two columns of Table 2.4 are normalized computation times. Here, *normalized* means the computation time is divided by the simulation duration (10 s) such a time of less than one is faster than real-time. These computation times are for the MATLAB implementation; see Section 2.5.4 for computation speeds in C++. The "motion time" is the computation time required just for motion prediction. The "total time" is the computation time required for the entire simulation, including geometric collision detection between the wheels and ground (which is four times as expensive as kinematic motion prediction). This data is plotted in Figure 2.10.

For a .04 s step size, motion prediction using our DAE method is 180× faster than using the unconstrained method; this is due to the elimination of terrain fitting, which requires an expensive nonlinear optimization at every time step. Computation times for the unconstrained method scale up with terrain roughness, due to the nonlinear optimizations requiring more iterations [46]. In contrast, computational cost for our non-iterative, stabilized DAE method is independent of roughness.

The DAE method errors for the .04 s step size are modest relative to the distance traveled: just 4 cm and 1.5° after driving 5 meters on rough terrain. But, using a .04 s step size is 4× faster than using a .01 s step size. For many applications, choosing a larger step size would be trading a marginal reduction in accuracy for a significant boost in computation speed.

In conclusion, our stabilized DAE method of kinematic motion prediction is more accurate and efficient than the traditional, unconstrained method for articulated WMRs on non-flat terrain. We acknowledge that, if terrain fitting were performed using our own method (in Section 2.2.4) vs. fminsearch, then the efficiency advantage of our DAE method would be reduced. Terrain fitting would be using the same wheel Jacobian information as the DAE, and Newton's method would usually converge in 1-2 steps. However, the accuracy advantage our DAE method would remain unchanged.

2.5.2 Comparison to Unconstrained Dynamics

Here we compare our DAE method for modeling WMR dynamics to the common ordinary differential equation (ODE) method. Instead of using constraints, contact and actuator forces are

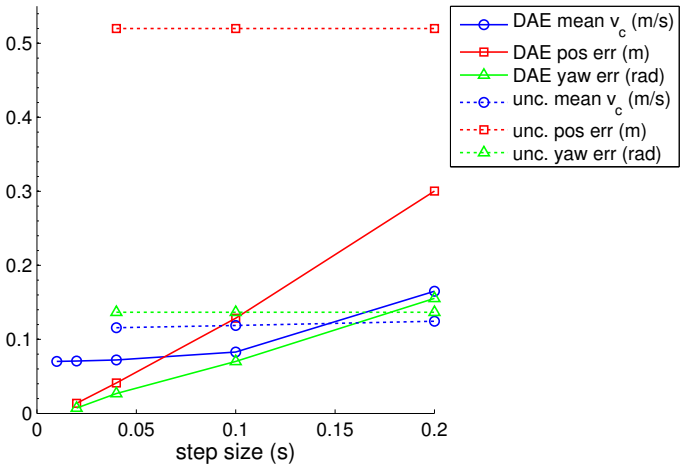


Figure 2.9: Mean contact point velocity, final position error, and final yaw error vs. step size for kinematic simulations of the Rocky 7 rover (from Table 2.4). Errors are smaller using our stabilized DAE method (solid) vs. the unconstrained method (dashed).

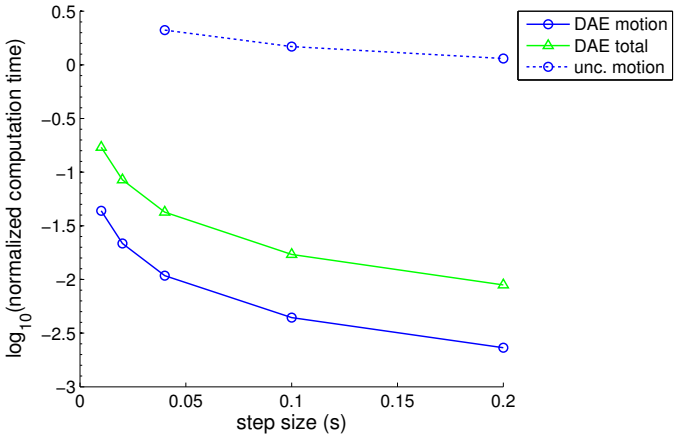


Figure 2.10: Normalized computation time vs. step size for kinematic simulations of the Rocky 7 rover (from Table 2.4). {-1,-2,-3} on the y-axis correspond to $\{10\times, 100\times, 1000\times\}$ faster than real-time.

added directly in ODE models (Section 2.4.1). Some have observed that this direct addition of forces makes the dynamics very stiff, such that the choice of contact model is limited or special integrators are required for stability [7][118]. Accordingly, we refer to this as the "stiff ODE" method.

Here we compare the two methods in a simple test of a LandTamer vehicle accelerating, traversing a ramp, then turning to the left and right (Figure 2.11). The LandTamer is a six-wheeled skid-steer platform made by PFM Manufacturing Inc. Frame information is provided in Table E.1 in Appendix E. We use the LandTamer in physical experiments in Section 3.4.2.

We chose an empirical wheel-ground contact model based on the Pacejka magic formula and the Nicolas-Comstock equations, as presented in [19]. Appendix C explains this contact model in detail. We chose a basic proportional controller for the actuator model, with a modest gain.

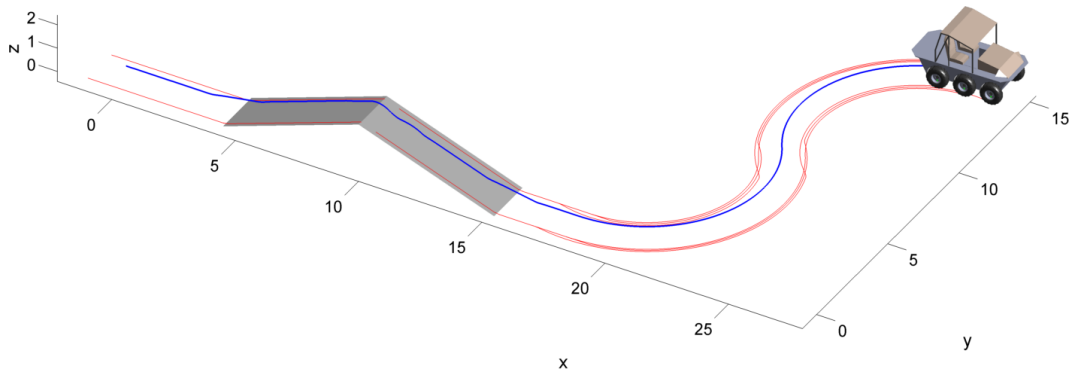


Figure 2.11: Animation screenshot for a dynamic simulation of the LandTamer

We performed the simulation using both our DAE method and the stiff ODE method. The terrain and trajectory are necessarily simple to keep computation times tractable for the stiff ODE method. At the 1 second mark, the LandTamer is commanded to abruptly accelerate from 1 to 2 m/s. Then, in the interval from 2-10 seconds it traverses a 1.5 m high, 20° ramp. At the 10 and 15 second marks the LandTamer is commanded to turn left and right respectively at 1 rad/s.

Final position error and computation times are provided in Table 2.5. For the DAE method we varied step size from .01 to .20 seconds. For the stiff ODE method we used MATLAB's built-in solvers for stiff systems: ode15s, ode23s, and ode23tb. According to MATLAB's documentation, ode15s is a variable-order, multistep solver for medium accuracy. ode23s and ode23tb are potentially more efficient at crude tolerances. Default settings were used, except the relative error tolerance (RelTol) was increased from 1e-3 to 1e-2 for faster computation, and the maximum step size (MaxStep) was set to .04 s. These solvers adaptively change the integration step size throughout the simulation. Step size vs. step number for the ode23tb solver is plotted in Figure 2.12. Mean step sizes chosen by each solver are presented in Table 2.5.

Table 2.5: LandTamer dynamic simulation errors and computation times

method	step size (s)	pos. diff. (m)	comp. time[1]
DAE	0.01	-	0.3204
	0.02	0.0041	0.1635
	0.04	0.0766	0.0903
	0.10	0.0053	0.0435
	0.20	0.3780	0.0281
ode15s	0.0025	0.0742	9.7647
ode23s	0.0114	0.0503	7.0499
ode23tb	0.0141	0.0508	1.5267

1. Computation times are for MATLAB implementation, and are normalized such that < 1.0 is faster than real-time.

Our DAE method takes fixed size steps, but the number of Newton's method iterations required for force-balance optimization varies (Section 2.4.3). Figure 2.13 plots the number of iterations required for dynamic simulation using the DAE method and .04 s step size. Initialization of the inputs to values at the previous time step (as explained in Section 2.4.4) means

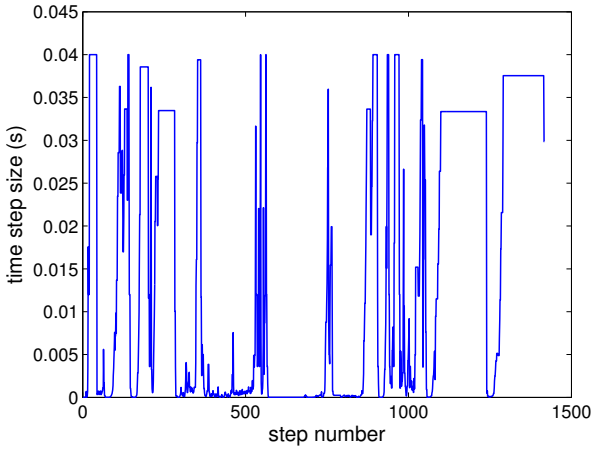


Figure 2.12: Integration step sizes for the dynamic LandTamer simulation, using the ode23tb solver. At times very small steps are required to maintain stability.

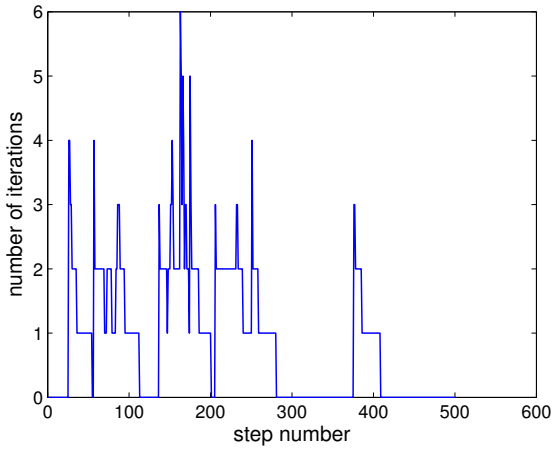


Figure 2.13: Number of Newton's method iterations required (for force-balance optimization) for the dynamic LandTamer simulation, using the DAE method and .04 s step size. During steady-state periods zero iterations are required.

that during steady-state periods zero iterations are required. The mean number of iterations required per time step was 0.87; fewer iterations are required when using a smoother wheel-ground contact model (Section 2.5.4).

In Table 2.5, final position difference is measured relative to the DAE simulation using the smallest (.01 s) step size. In this test, we do not claim that using a DAE is more correct than using a stiff ODE and adaptive solver; rather we claim that both methods make similar predictions. The position difference for all stiff ODE solvers is less than 8 cm after driving 40 m (0.2%). Modulating the step size for the DAE method can affect accuracy; for a more detailed analysis of this see Section 2.5.4, in which results are computed for multiple trials.

To further illustrate the closeness of DAE and stiff ODE motion predictions, slip ratio and angle vs. time are plotted for the LandTamer's middle-left wheel in Figure 2.14. Plots for all wheels are provided in Figures F.1 and F.2 in Appendix F. Note that the simulation output matches expectations for the chosen tire model. Slip ratios indicate that the vehicle slips longitudinally in the direction of gravity when traversing the ramp (from 2-10 seconds). Slip angles indicate that the vehicle slips laterally away from the center of curvature when turning (from 10-20 seconds) due to centripetal acceleration. There is some jitter in the stiff ODE simulation visible at the 6 second mark in the slip ratio plots. Without adaptively reducing the step size, this jitter would become severe instability.

Computation times in Table 2.5 are normalized, or divided by the simulation duration (20 s) such that times less than one are faster than real-time. Computation using the DAE method, which takes fixed size steps, is much faster than using a stiff ODE and adaptive solver. In this test, for a step size of .04 s, the DAE method was 17-108$\times$ faster than the stiff ODE method, depending on the solver used. Simulation speeds using a C++ implementation of our DAE dynamics formulation are benchmarked in Section 2.5.4.

In conclusion, our DAE-based WMR dynamics formulation provides equal fidelity to the

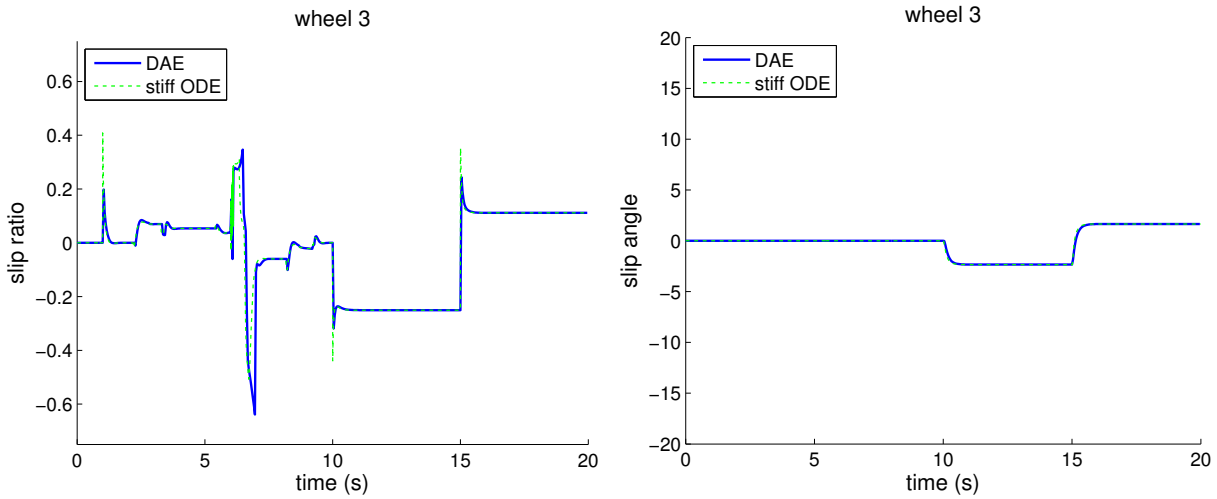


Figure 2.14: Slip ratio vs. time (Left) and slip angle vs. time (Right) for the LandTamer's middle-left wheel during a dynamic simulation. Solid lines represent DAE method output, dashed lines represent stiff ODE method output. See plots for all wheels in Figures F.1 and F.2.

common ODE-based formulation; however, our formulation is stable even for large step sizes, which greatly increases computation speed.

2.5.3 Prediction of Extreme Phenomena

Section 2.5.2 demonstrated motion prediction for conservative maneuvers, but our dynamics formulation can also predict the extreme phenomena that occur during aggressive maneuvers. Here we demonstrate the prediction of *loss of traction* and *rollover*. We use the same skid-steer LandTamer platform and empirical wheel-ground contact model used in Section 2.5.2.

Loss of traction. Loss of traction is observed experimentally at high slip ratios and angles. This can be seen in Figure 2.15, which presents the results of a study by Brach and Brach [18]; force begins to *decrease* as slip ratio increases above a critical value (approximately 0.2). Modern automobiles are equipped with anti-lock braking systems (ABS) and electronic stability control to address this phenomenon; these systems keep slip below the critical value to improve stopping distance and controllability.

First we demonstrate the capability of our dynamics formulation to predict this phenomenon when abruptly accelerating/decelerating while driving straight. The longitudinal force and slip ratio vs. time for this test are plotted in Figure 2.16. At the 1 second mark the LandTamer accelerates from 0.5 to 5 m/s, and at the 3 second mark it decelerates back to 0.5 m/s. Note that the coefficient of friction was lowered to 0.6 to simulate a low traction surface.

Longitudinal force vs. slip ratio is plotted in Figure 2.17. Notice that our simulation output matches the experimental results in Figure 2.15; The *magnitude* of longitudinal force decreases as the *magnitude* of slip ratio increases above a critical value. Positive slip ratios correspond to acceleration and negative to deceleration (or braking).

Our dynamics formulation enforces an even more complex relationship between force and slip when the LandTamer accelerates/decelerates while turning. To demonstrate, we repeated

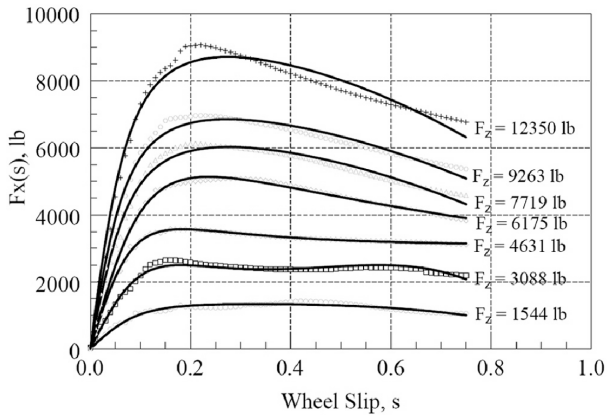


Figure 2.15: Longitudinal contact force vs. slip ratio for a pneumatic tire, based on a study by Brach and Brach [18].

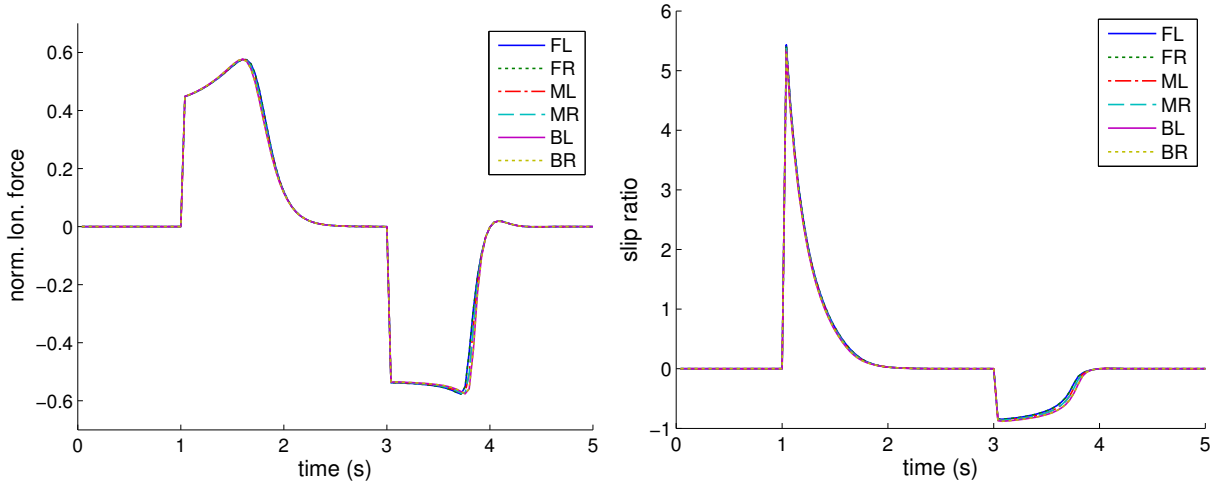


Figure 2.16: Normalized longitudinal force vs. time (Left) and slip ratio vs. time (Right) as the LandTamer accelerates/decelerates while driving straight. For all wheels: Front Left, Front Right, Middle Left, Middle Right, Back Left, Back Right. *Normalized* longitudinal force is the ratio of longitudinal force to normal force (f_x/f_z).

the test while commanding the LandTamer to turn at 2 rad/s at full speed. A severe loss of yaw stability occurs, as seen in Figure 2.18. Figure 2.19 plots the longitudinal force vs. slip ratio and angle. Notice that as slip angle increases, longitudinal force decreases (as a component of the contact force is directed laterally). At every time step, cost in the force-balance optimization was successfully minimized to zero (within a numerical tolerance), indicating that the specified wheel-ground contact model was satisfied.

Our enhanced kinematic model (Section 2.3) cannot predict this type of extreme slip, it can only enforce a linear force-slip relationship at the body-level. Open Dynamics Engine also cannot enforce nonlinear wheel-ground contact models like this. OpenDE uses constraints like our formulation but not our force-balance optimization technique (Section 2.4.3); it can only enforce a bounded, linear force-slip relationship at the wheel-ground contacts, or an approximation of

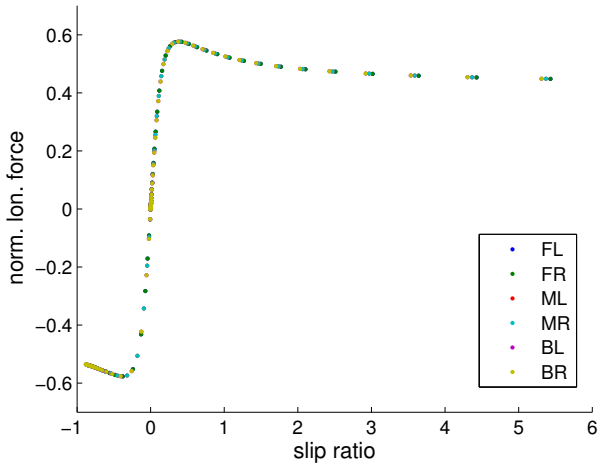


Figure 2.17: Normalized longitudinal force (f_x/f_z) vs. slip ratio as the LandTamer accelerates/decelerates while driving straight.

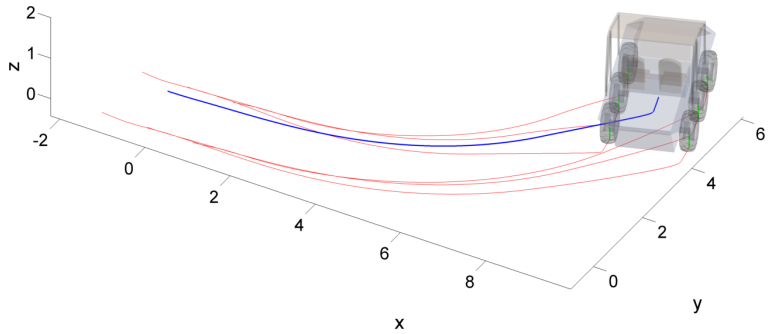


Figure 2.18: Animation screenshot for a dynamic simulation of the LandTamer accelerating/decelerating while turning. Loss of yaw stability causes the vehicle to turn sharply.

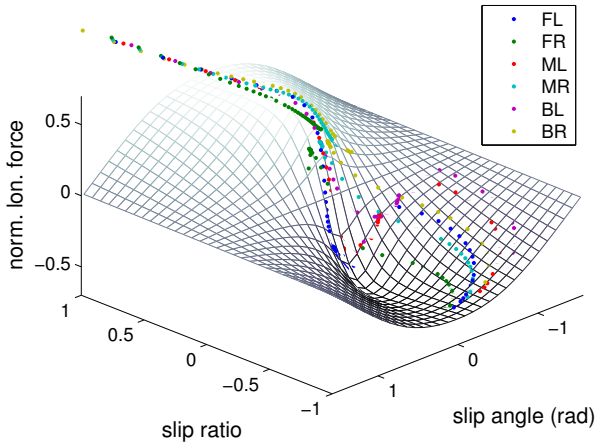


Figure 2.19: Normalized longitudinal force (f_x/f_z) vs. slip ratio and angle as the LandTamer accelerates/decelerates while turning. The dots represent the simulation output for all six wheels, and the mesh represents the force-slip relationship for the specified wheel-ground contact model over all slip ratios/angles.

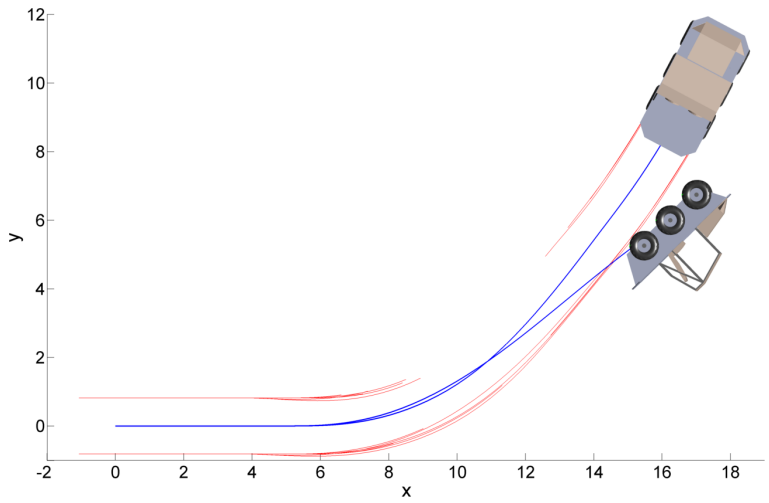


Figure 2.20: Overlaid animation screenshots for two dynamic simulations of the LandTamer executing a high-speed turn. Blue lines trace body frame position, red lines trace wheel-ground contact point positions. As seen by the break in the red lines, both LandTamer models experience liftoff, but only the model with a raised cg height rolls over.

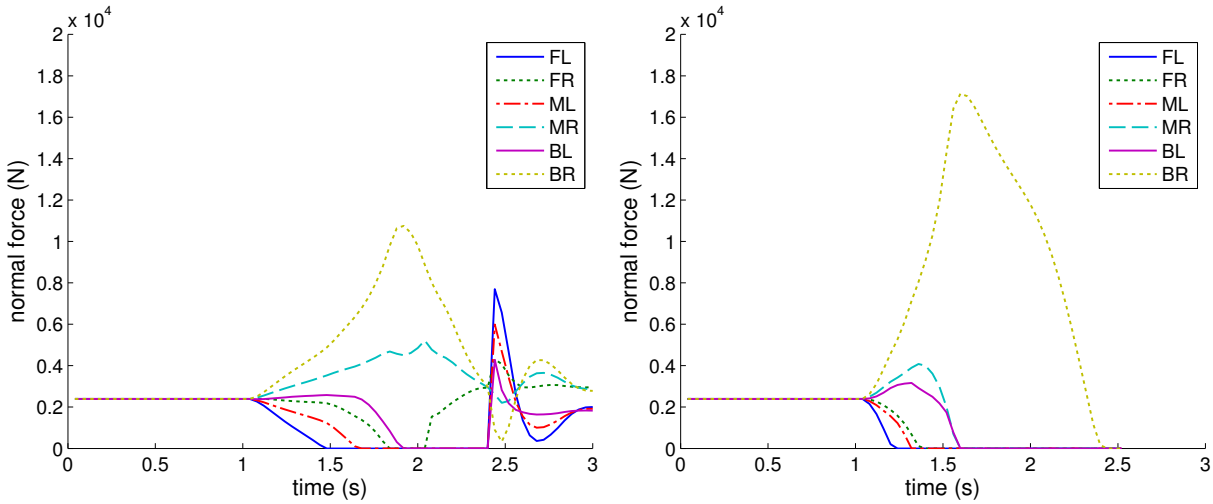


Figure 2.21: Normal forces vs. time for each wheel during dynamic simulations of the LandTamer executing a high speed turn. Liftoff occurs when normal force is zero. For the baseline cg height model (Left) four wheels liftoff but eventually reestablish contact with the ground. For the raised cg height model (Right) the vehicle rolls over, pivoting about the back right wheel

Coulomb's law of friction.

Rollover. WMRs are susceptible to rollover due to gravitational and inertial force. Rollover risk is greatest on steep slopes or when making aggressive turns. Added payloads can also increase rollover risk by raising the cg.

Ordinary kinematic models provide no warning about rollover. As explained in Section 2.3, our enhanced kinematic model can indicate when liftoff is imminent due to gravity and some types of acceleration. Our full dynamic model can accurately predict liftoff due to any source,

and predict motion after liftoff to determine if rollover will occur. To demonstrate, we induce liftoff and rollover in a dynamic simulation of the LandTamer vehicle. The LandTamer is commanded to drive straight at 5 m/s, then turn sharply at 7 rad/s at the 1 second mark. The simulation is repeated for two LandTamer models: one with a baseline cg height of 0.83 m, and one with a raised cg height of 1.25 m.

Overlaid animation screenshots for the two simulations are shown in Figure 2.20. Normal forces vs. time for each wheel are plotted in Figure 2.21. Both LandTamer models experience liftoff for some wheels, but only the model with a raised cg height ultimately rolls over.

In conclusion, our DAE dynamics formulation with force-balance optimization is capable of enforcing realistic, nonlinear models of wheel-terrain interaction. We predict loss of traction phenomena that kinematic and even Open Dynamics Engine models cannot predict. We also predict liftoff and rollover. Accurate prediction of these extreme phenomena is essential for model predictive control of WMRs during aggressive maneuvers.

2.5.4 Computation Speed Benchmarking

Previous sections have presented faster than real-time WMR simulations using MATLAB implementations of our kinematic and dynamic model formulations. Here we demonstrate even faster computation speeds using C++ implementations. We also assess the effect of integration step size on simulation accuracy.

Open Dynamics Engine is a popular, open-source physics engine which many have used for WMR modeling [52][75][44][101][32][42]. In Section 2.5.3 we demonstrated the capability of our dynamics formulation to enforce more realistic, nonlinear models of wheel-terrain interaction than OpenDE. Here we demonstrate that such models can be enforced while achieving comparable computation speeds to OpenDE.

We benchmark all models in a simulation of the Zoë rover traversing rough terrain. The Zoë rover was built to autonomously survey the distribution of microbiological life in the Atacama desert [132]. Zoë has two articulated axles; each has a passive steering degree of freedom controlled indirectly by differential left/right wheel speeds. The rear axle also has a roll degree of freedom to keep the wheels in contact on uneven terrain [131]. Frame information is provided in Table E.2 in Appendix E. We use the Zoë rover in 3D odometry experiments in Section 3.4.3.

In this test, Zoë was commanded to drive straight at 1 m/s over a randomly-generated rough terrain surface with a root mean square (RMS) height of 0.15 m and a correlation length of 0.8 m (Figure 2.22). Table 2.6 presents computation speed results for different model types. All results are averaged over ten trials, each with a unique rough terrain surface. All simulations were run on a desktop computer with a 3.40 GHz Intel processor, using our C++ software library and double precision. Note that computation speeds in the table are normalized (divided by simulation duration) and scaled such that times less than one are over 1000$\times$ faster than real-time. Computation speed data is also plotted in Figure 2.23.

The columns of Table 2.6 correspond to the following model types:

- *kinematic*: Our kinematic model formulation (Section 2.2)
- *erp/cfm*: Our dynamics formulation (Section 2.4), but using Open Dynamics Engine's contact method instead of force-balance optimization. OpenDE uses error reduction pa-

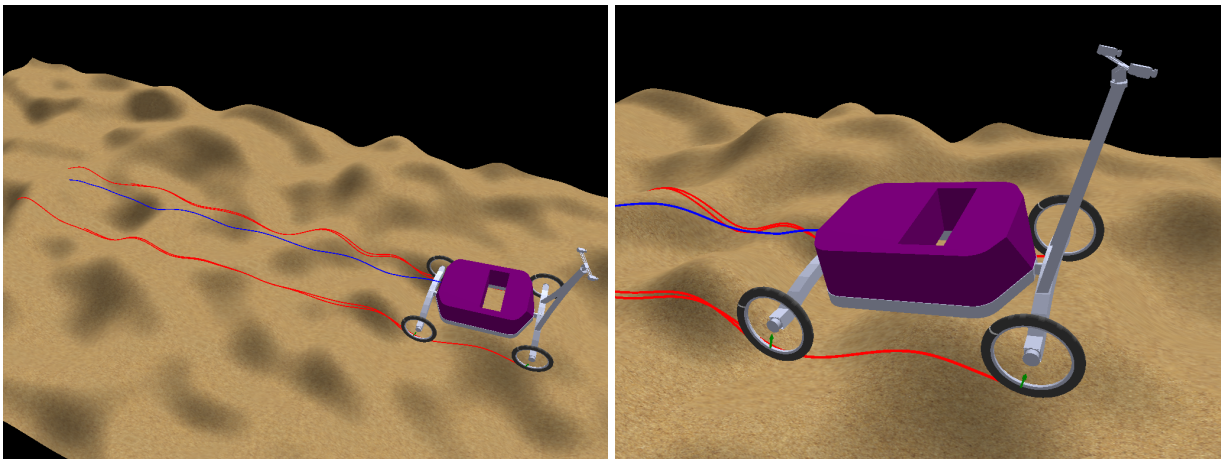


Figure 2.22: Animation screenshots of the simulation used to benchmark computation speeds: the Zoë rover traversing rough terrain.

Table 2.6: Normalized computation times for the Zoë rover benchmark

step size (s)	kinematic	erp/cfm	piecewise-linear	Pacejka	terramech.-based	OpenDE
0.01	0.9840	1.7990	3.1380	4.3660	3.6090	2.9060
0.02	0.4690	0.9040	1.5620	2.2030	1.8740	1.4380
0.04	0.2190	0.4540	0.7800	1.1410	0.9530	0.7340
0.10	0.0780	0.1910	0.3120	0.5000	0.3600	0.2960

All computation times are normalized and scaled ($\times$1e-3) such that < 1.0 is over 1000$\times$ faster than real-time. Model types are explained in the text.

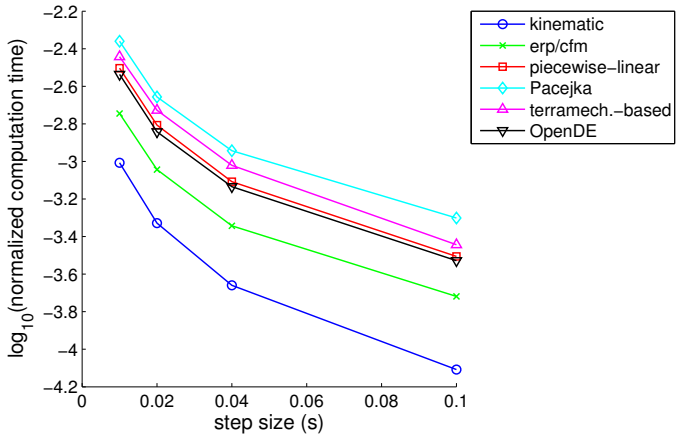


Figure 2.23: Normalized computation times for the Zoë rover benchmark

rameters (erp) and constraint force mixing (cfm) to model spring/damper effects at contact points (see [116] Sections 3.8-3.9)

- *piecewise-linear*: Our dynamics formulation, using a piecewise-linear wheel-ground contact model. Force is linearly proportional to slip velocity up to a limit, above which force is constant.
- *Pacejka*: Our dynamics formulation, using the Pacejka/Nicolas-Comstock model from [19] for wheel-ground contact.
- *terramech.-based*: Our dynamics formulation, using a terramechanics-based contact model from [52]. A precomputed lookup table is used to speed up computation.
- *OpenDE*. A dynamics model entirely constructed and simulated using Open Dynamics Engine, except using our (faster) collision detection engine instead of the one packaged with OpenDE.

More information about the wheel-ground contact models used is provided in Appendix C.

When using our kinematic model formulation, the simulation runs nearly 10K$\times$ faster than real-time for a .10 s step size. When using our dynamic model formulation and Open Dynamics Engine's erp/cfm method for contact modeling, the simulation runs over 1K$\times$ faster than real-time for step sizes of .02 s or larger. Computation time is reduced by 40% relative to the OpenDE simulation, due to our use of a minimal state space and recursive algorithms (which are ideal for articulated mechanisms with single DOF joints). We acknowledge that the enforcement of bounds on contact forces was not required for this test (i.e. $f_z \geq 0$, $f_x \leq \mu_x f_z$, $f_y \leq \mu_y f_z$). If enforcement of these bounds was required, solving the linear complementarity problem (LCP) would have taken longer, increasing the computation time for both models.

When using our dynamic model formulation with force-balance optimization, the simulation runs slightly slower than when using Open Dynamics Engine, but still on the order of 1K$\times$ faster than real-time for step sizes of .04 s or larger. In general, computation time increases with the complexity of the specified wheel-ground contact model. The terramechanics-based model is actually more complex than the Pacejka model, but runs faster for two reasons. First, we use a precomputed lookup table; this is possible for any wheel-ground contact model given sufficient memory. Second, the terramechanics-model is smoother such that, on average, fewer iterations of Newton's method are required during force-balance optimization (Table 2.7).

Ishigami et al. also performed full dynamic simulation of an articulated rover using this same terramechanics model, but their simulations ran three times slower than real-time (on a 1.66 GHz processor) [53]. Our simulations run at least *three orders of magnitude* faster, making their planning algorithm computationally tractable.

Increasing the integration step size can reduce computation time, but may also affect accuracy. Table 2.7 provides final position and yaw errors for dynamic simulations using different step sizes. Here, error is measured relative to the predicted position/yaw using the smallest (.01 s) step size. Results are provided for dynamic models using three wheel-ground contact models already discussed. For all three, mean final position error is 5 cm or less and mean final yaw error is less than 1° after driving 10 m using a .10 s step size. These results indicate that our DAE dynamics formulation can achieve adequate accuracy for MPC/MPP even for large step sizes.

Table 2.7 also provides the mean number of Newton's method iterations required per step.

The choice of wheel-ground contact model has the biggest impact on the number of iterations, but the number also increases with step-size for the nonlinear models. We tried the force-balance stabilization technique described in Section 2.4.4, in which a single Newton's method step is taken per time step. This technique only reduced computation time by 10-15%. The computational benefit is small because our caching of intermediate variables makes subsequent iterations of Newton's method cheap.

Table 2.7: Final pose errors and iterations required for Zoë rover benchmark

	piecewise-linear			Pacejka			terramechanics-based		
step size (s)	pos. err. (m)[1]	yaw err. (rad)[2]	iter[3]	pos. err. (m)	yaw err. (rad)	iter	pos. err. (m)	yaw err. (rad)	iter
0.01	-	-	0.9990	-	-	1.7364	-	-	1.4118
0.02	0.0039	0.0003	0.9980	0.0037	0.0006	1.8663	0.0085	0.0020	1.6525
0.04	0.0126	0.0009	0.9960	0.0109	0.0014	2.0100	0.0226	0.0043	1.9657
0.10	0.0401	0.0039	0.9901	0.0292	0.0038	2.3624	0.0513	0.0128	2.3149

All results are averaged over 10 trials.

1. Euclidean distance error of final position, relative to simulation using the smallest (.01 s) step size
2. absolute error of final yaw
3. mean number of Newton's method iterations required per step for force-balance optimization

In conclusion, our DAE dynamics formulation is capable of simulating articulated rovers on rough terrain *3-4 orders of magnitude* faster than real-time. Our formulation provides higher-fidelity modeling of wheel-terrain interaction than Open Dynamics Engine, but at comparable computation speed. These speeds are already sufficient for many MPC/MPP planning applications, but additional speedup may be possible.

Computational bottlenecks include solving linear systems, and calculating the gradient and Hessian in the force-balance optimization. A better optimized matrix library could potentially speed up these operations. In addition, our software implementation does not yet fully exploit vectorization or parallelization opportunities. In model predictive planning, the simulation task is embarrassingly parallel in the number of candidate trajectories. Information reuse can also be exploited for overlapping trajectories [71].

Chapter 3

Model Calibration

This chapter presents a novel method to calibrate the kinematic and dynamic model formulations presented in Chapter 2. Section 3.1 provides an overview of our Integrated Prediction Error Minimization (IPEM) method and discusses related work. Section 3.2 explains the theory and implementation of IPEM for systematic and stochastic models. Section 3.3 applies IPEM to WMR model calibration. Finally, Section 3.4 assesses the predictive accuracy of calibrated WMR models in physical experiments on multiple platforms and terrain types.

3.1 Related Work on Model Identification

When modeling complex physical systems, sufficient detail in the formulation alone does not guarantee accuracy; the model parameters must also be correctly identified. Accordingly, we have developed an Integrated Prediction Error Minimization (IPEM) method to calibrate our WMR models, or any dynamical model. This section provides a brief overview of IPEM (Section 3.1.1), then discusses related work on model identification in general (Section 3.1.2) and on WMR model identification in particular (Section 3.1.3).

3.1.1 IPEM Overview

System dynamics are commonly expressed as a differential equation (DE):

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}) \tag{3.1}$$

where $\mathbf{x}$ is the state, $\mathbf{u}$ is the input vector, and $\mathbf{p}$ is the vector of parameters to be identified. The function $\mathbf{f}()$ is often nonlinear. Model identification is achieved through the minimization of an objective function of residuals. The traditional approach is to form residuals directly from the differential equation:

$$\mathbf{r}(t) = \dot{\mathbf{x}}_{meas}(t) - \dot{\mathbf{x}}_{pred}(t) \tag{3.2}$$

This DE residual compares *instantaneous* measured (*meas*) vs. predicted (*pred*) output.

This requires observations of $\dot{\mathbf{x}}$, which often cannot be measured directly; instead measurements of state ($\mathbf{x}$) are numerically differentiated with respect to time. For example, the

Springer Handbook of Robotics describes the estimation of manipulator inertial parameters this way, which requires the double-differentiation of joint angle measurements [114].

In contrast to the traditional approach, we form IPEM residuals by integrating the dynamics over an interval:

$$\mathbf{r}(t) = \mathbf{x}_{meas}(t) - \mathbf{x}_{pred}(t) \tag{3.3}$$

$$\mathbf{x}_{pred}(t) = \mathbf{x}_{meas}(t_0) + \int_{t_0}^{t} \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \mathbf{p}) d\tau \tag{3.4}$$

Where t_0 and t denote the start and end of the interval, respectively. In effect we integrate the prediction (i.e. forward simulate the model) rather than differentiate the measurement, which has several advantages:

Reduced sensing requirements. With IPEM, measurements can be temporally and spatially sparse. The numerical time differentiation of measurements for the traditional approach requires high-frequency data and accurate time stamps. IPEM only requires measurements at t_0 and t, which can be seconds apart. Because the signal (i.e. change in state) is larger over longer intervals, there is less sensitivity to the noise of time stamp errors. Obtaining high-frequency data also requires that measurements be available spatially throughout the trajectory. With IPEM one can execute varied trajectories between just two surveyed points in state space.

Better observability. With IPEM, the compounding effects of integrated error become a virtue. Small errors accumulate until they become observable or easier to disambiguate. For example, for a WMR, an error in predicted angular velocity has no instantaneous effect on predicted linear velocity, but it has an increasing effect on predicted position with distance traveled. A gyro could observe the angular error directly, but even GPS can observe it with IPEM.

Longer prediction horizon. IPEM optimizes predictive accuracy for a longer horizon ($t-t_0$) than the traditional approach. Calibration to DE residuals may provide good one-step ahead predictions, but poor predictions over longer intervals. This is especially true for stochastic model calibration (Section 3.4.1). With IPEM one can specify the horizon to optimize for, according to the needs of the application.

The tradeoff to using IPEM is added complexity and computation. Despite its elegance, exploiting the principle of integrated predictions in practice requires some effort.

3.1.2 Related Work on Model Identification

Naming our method was challenging. Classical references distinguish between "equation error" (EE) and "output error" (OE) models by their assumption of noise either in the process equation or the output measurement [79][117]. Our method assumes and characterizes both types of noise.

Nicolao [91] makes another distinction between EE and OE approaches to model identification. When forming EE residuals, the output at time k is predicted using a sequence of past input and output values (at time steps $k-1, \ldots, k-n$). In OE residuals, the output at time k is calculated by forward simulating multiple time steps given only the inputs and initial conditions.

Nicolao restricts "prediction" to mean *one-step ahead* prediction, and calls many-steps ahead prediction "simulation." Note that residuals based on one-step ahead prediction are equivalent

to those based directly on the differential equation (3.2), if measurements are numerically time-differentiated. This equivalence becomes clear with some manipulation of the system differential equation (3.1) expressed in discrete time:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{p})\Delta t \tag{3.5}$$

$$\frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\Delta t} = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{p}) \tag{3.6}$$

The left hand side of (3.6) is a numerical time derivative computed via forward difference.

Using Nicolao's terminology, EE/prediction approaches are more traditionally used for WMR model identification, whereas IPEM is more similar to the OE/simulation approach. Note, however, that we do not use "prediction" so restrictively; in this document "prediction" often refers to forward simulation over an interval.

The advantage of EE/prediction is that, if the function $\mathbf{f}()$ is linear in the parameters ($\mathbf{p}$) then so are the residuals, and $\mathbf{p}$ can be easily computed by linear least-squares. In OE/simulation the residuals are *nonlinear* in the parameters (with the associated problems of convergence, initialization, local minima, etc.), but the identified model will be suitable for simulation over a longer horizon [129].

Several have studied the benefits of forming residuals over longer intervals. Farina and Piroddi [35] compared multi-step prediction error minimization (PEM) to simulation error minimization (SEM) and related them both to OE. Jørgensen and Jørgensen [58] compared prediction error modeling criteria (maximum likelihood vs. least squares), and showed that multi-step PEM is effective at identifying models for MPC [57]. We call our method "integrated" rather than "multi-step" because we derive the mathematics in continuous time. Of course, conversion to discrete time is straightforward and necessary for numerical implementation.

While many in robotics have studied the identification of systematic (or deterministic) dynamics, there is less related work on identifying *stochastic* dynamics. The most closely related work is on Kalman filter (KF) tuning. Kalman filters propagate the uncertainty in system state over time, accounting for measurement and process noise. Kalman filters assume Gaussian noise, and so can fully represent measurement and process noise by covariance matrices, typically denoted R and Q. The tuning of Q is most critical, as R can usually be determined from a priori knowledge of the sensor characteristics [104].

Various algorithms have been proposed for Kalman filter tuning. For example, Åkesson et al. [3] solve for noise covariances using linear least squares, but only for linear time-invariant systems. Others have used fuzzy logic [98], a simplex algorithm [97], and neural networks [72]. Some focused on the choice of objective function rather than the optimization algorithm. Abbeel et al. [1] provide several options: maximizing the joint likelihood of all data (if the full state is observable), or maximizing just the prediction or measurement likelihood. In experimental results on a LAGR platform, they showed improved accuracy of KF state and state covariance estimates over manual tuning. Saha et al. [104] provide metrics based on innovation covariance (S) to choose combinations of parameters that balance robustness and sensitivity.

3.1.3 Related Work on WMR Model Identification

There are numerous parameters to identify in WMR models:

- dimensions: wheel radii, wheelbase (distance between front & rear wheels), track (distance between left & right wheels), etc.
- mass properties: mass, center of mass, moment of inertia for each frame
- wheel-ground contact model constants: soil properties, tire properties
- actuator model constants: torque-speed characteristics, limits

This section summarizes related work on WMR model identification. Most publications address the identification of dimensions for *odometry*. In this document, odometry refers to the dead-reckoning of robot position from measurements of wheel velocity and other joint angles/rates (e.g. steer angles), if applicable.

There are many ways to identify WMR model parameters. One may use nominal values (e.g. CAD model dimensions) or take manual measurements. One may also analyze subsystems in isolation. For example, Goel et al. [40] calibrated encoders with a more precise tachometer while his robot sat on a box and its wheels spun freely. Ishigami et al. [52] and Ding et al. [31] determined soil parameters in terramechanics equations by driving a single wheel in an instrumented testbed. Commercial rigs equipped with dynamometers can be used to test and characterize actuators. Calibrating subsystems independently can be laborious and inaccurate; the vehicle model may not match experimental data even when subsystem models do, due to unmodeled dependencies between the subsystems.

Some have proposed full vehicle calibration routines. For example, Borenstein and Feng [17] developed the "UMBmark" test, in which a differential drive robot is driven in a square path to calibrate wheel diameters and track. Iagnemma [49] presented a simple maneuver for terramechanics identification in which one wheel on a rover is driven slower than the others.

The best way to calibrate WMR model parameters is *online* during normal operation. This is more convenient and accurate than offline methods, as it enables adaptation to changing conditions like a terrain transition or flat tire. Some have provided online odometry calibration methods that require frequent measurements. For example, Roy and Thrun [102] calibrate systematic translation and rotation errors by maximizing the likelihood of laser scans. Rudolph [103] calibrated encoder scale factor errors and track width using a rate gyroscope as a complementary sensor.

Others provided odometry calibration methods that require less frequent measurements, i.e. just the initial and final pose in each trajectory. Martinelli [83] estimates two systematic error parameters from "observables" for repetitions of a straight back and forth trajectory. Antonelli et al. [5] solve for four odometry parameters via linear least-squares. They use numerical conditioning of the data to guide the choice of test trajectories, but require no special shape. Kelly [61] provided a "fast and easy" calibration procedure based on his derivation of linearized error propagation in odometry [60]. Several have combined odometry calibration with other tasks, for example: laser scanner calibration [22], camera calibration [6][70], and simultaneous localization and mapping (SLAM) [73].

Even after calibrating out systematic errors, some nonsystematic error will always remain due to random disturbances and sensor noise. Even in controlled experiments, repeated trajectories will not match exactly. For example, Sohl and Jain [118] observed wide variations between runs in experimental mobility tests on the MER rovers. Collision or other failures may occur if a WMR's planner/controller does not account for this uncertainty (Figure 3.1).

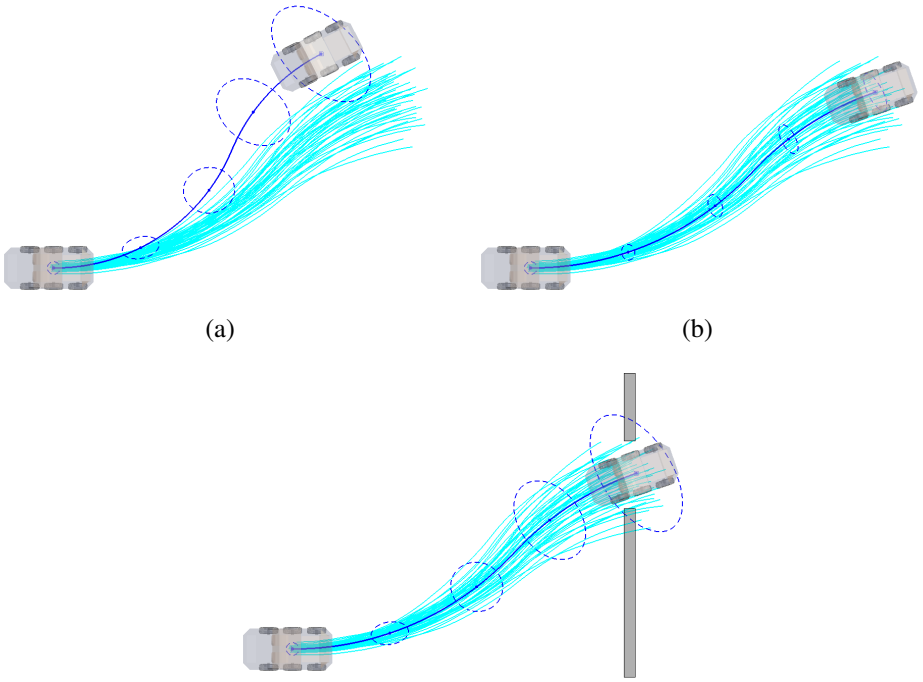


Figure 3.1: Monte-Carlo simulation of a WMR trajectory with noise. Individual trajectories vary due to noise in the measurements of initial conditions and random disturbances. The dark, solid lines represent predicted mean trajectories, and the dashed-line ellipses represent predictions of uncertainty. Uncertainty must be accounted for in planning/control to avoid collision. (a) results with poor systematic calibration. (b) with poor stochastic calibration. (c) with good systematic and stochastic calibration.

Some have characterized nondeterministic or *stochastic* odometry error. Chong and Kleeman [27] first derived closed form expressions for the sums of noise over particular paths (circular arc, straight line, rotation about axle center). Martinelli simultaneously estimated two nonsystematic error parameters along with the two systematic parameters in [83]. In [84], Martinelli updated his approach to estimate these parameters online via an augmented Kalman filter and observable filter. Kelly [61] identified the process noise matrix Q in a linear variance equation without requiring the repetition of trajectories.

Calibrated error propagation models can be used in probabilistic WMR position estimation systems (EKF, particle filter, etc.), and probabilistic motion planners (such as [41][13]).

3.1.4 Comparison of Related Work on Model Identification

Table 3.1 summarizes the features of related work on WMR model calibration. The columns are:

- Model: what type(s) of WMR model can be calibrated? (general)
- Any trajectory: no special trajectory or routine required (convenient)
- Limited sensing: ground truth can be sparse (convenient)

Table 3.1: Comparison of related work on WMR model calibration

Publication	Model	Any traject.	Limited sens.	Online	Stoch.	Simultaneous task
Borenstein and Feng 1996	diff. drive		X			
Chong and Kleeman 1997	diff. drive		X		X	
Roy and Thrun 1999	unicycle			X		
Goel et al. 1999	diff. drive					
Martinelli 2002	unicycle		X		X	
Rudolph 2003	diff. drive	X		X		
Kelly 2004	unicycle	X	X		X	
Antonelli et al. 2005	diff. drive	X	X			
Martinelli et al. 2007	diff. drive, synch. drive	X		X	X	localization
Censi et al. 2008	diff. drive	X				laser cal.
Antonelli et al. 2010	diff. drive	X	X			camera cal.
Kümmerle et al. 2012	diff. drive	X		X		SLAM
Kim et al. 2013	diff. drive	X				camera cal.
Seegmiller et al. 2013	3D unicycle	X	X	X	X	slip, sensor pos., powertrain cal.
Seegmiller and Kelly 2014, (this thesis)	3D kinematic, dynamic	X	X	X	X	

- Online: procedure can run online to adapt to changing conditions (online)
- Stochastic: characterizes nonsystematic error (evaluative)
- Simultaneous task: performs a task in addition to odometry calibration

The first five features map to the requirements listed in Section 1.1.2. The "unicycle" model type in the table refers to a simplified WMR model with only two instantaneous degrees of freedom: translation along a forward axis, and rotation about a vertical axis. Many WMR model types (e.g. differential drive, skid-steer) can be reduced to a unicycle model. Table 3.1 alone does not fully convey our contribution; we not only meet all requirements, but meet them in different ways than prior work.

The major contributions of this thesis to WMR model calibration include the following:

- This thesis provides a novel Integrated Prediction Error Minimization (IPEM) method for model calibration. Related work on odometry calibration is mostly limited to 2D and differential drive platforms. In contrast, our IPEM method is completely general, i.e. it can be applied to any vehicle model (or any differential equation). The IPEM method also meets the requirements of being convenient, online, and evaluative in novel ways. In particular, it uses unified systematic and stochastic error models and calibrates both simultaneously in a single filter. While there is precedent for using integrated prediction errors in WMR calibration [83][5], we uniquely analyze the benefits over calibrating to instantaneous output errors.
- This thesis applies IPEM to the calibration of the high-fidelity WMR models presented in Chapter 2 (i.e. articulated, 3D, kinematic and dynamic). Our early work on IPEM used much simpler models. Suspension deflections were ignored, and instantaneous motion was restricted to 3 DOF in a plane tangent to the terrain surface. Closed-form 2D kinematics

were used to map wheel velocities to planar motion. Body-level slip was parametrized by an ad hoc polynomial of velocity and gravity terms, a less principled parametrization than in Section 2.3.

- This thesis provides unprecedented validation of both our formulation and calibration methods in *physical experiments*. We have tested on numerous vehicles and terrain types; only a subset of our results are presented in this document. To our knowledge, no prior work has attempted such an extensive comparison of kinematic vs. dynamic models (Section 3.4.2) or evaluated 3D odometry over such long distances (Section 3.4.3).

IPEM was developed in collaboration with Forrest Rogers-Marcovitz and Alonzo Kelly, and it builds on Kelly's prior derivation of linearized error propagation in odometry [60]. Early results (using simple vehicle models) are published in [106][107] and most comprehensively in [108].

The author made significant contributions to the development of IPEM, including the derivation of expressions for residual covariances that correctly account for both measurement and process noise. All contributions regarding IPEM and high-fidelity models were made independently by the author. Some results using high-fidelity models are published in [111].

3.2 Integrated Prediction Error Minimization

This section explains the theory and implementation of our Integrated Prediction Error Minimization (IPEM) method, in the course of defining all necessary variables. Section 3.2.1 defines a measurement ($\mathbf{z}$), prediction ($\mathbf{h}$), measurement covariance (R), and measurement Jacobian with respect to parameters (H) for systematic model calibration. Section 3.2.2 defines corresponding variables for stochastic model calibration. Systematic and stochastic variables are distinguished by the subscripts μ and σ respectively, and are compared in Section 3.2.3. Finally, Section 3.2.4 presents offline and online IPEM algorithms.

3.2.1 Systematic Variables

Here we define variables for systematic model calibration (or systematic calibration for short). By "systematic model" we refer to the deterministic (often nonlinear) system dynamics expressed in (3.1) and (3.4). Calibration of the parameter vector $\mathbf{p}$ eliminates systematic errors from model predictions.

Systematic measurement. The measurement for systematic calibration is:

$$\mathbf{z}_\mu = \mathbf{r}(t) \tag{3.7}$$

where $\mathbf{r}(t)$ is the residual defined in equation (3.3). Computing the predicted final state $\mathbf{x}_{pred}(t)$ in $\mathbf{r}(t)$ requires numerical integration of the system differential equation (i.e. forward simulation) from time t_0 to t, per (3.4). This makes $\mathbf{x}_{pred}(t)$ not merely a function but a functional; it maps the values of the function $\mathbf{u}(\tau)$ over an entire interval to another value.

The ideal interval length is application dependent. In general, accurate prediction over longer intervals is desirable, but can be difficult. If there is mismatch between the chosen model formulation and the true process, the predictive accuracy for longer intervals may be poor. Dividing

training data into longer intervals yields fewer independent residuals; calibration to overlapping intervals is permissible, but residuals will be correlated. When calibrating a model for MPP, one could set the interval length to the horizon used by the planner. Alternatively, Duong and Landau [33] suggest a minimum necessary test horizon for the validation of linear models based on the cross-correlation of inputs and outputs.

For simplicity, our presentation assumes the full state vector $\mathbf{x}$ is observable, but this method can be modified if only partial measurements of state are available, or if measurements are a function of state. While only intermittent measurements of state are required (at times t_0 and t), inputs $\mathbf{u}$ must be sampled at sufficiently high frequency throughout the interval. Likewise, the numerical integration (or simulation) time step size must be sufficiently small.

Finally, it is conceptually simplest to think of t_0 as the current time and t as some future time, as though predicting a trajectory given a planned sequence of inputs. However, in practice replanning often occurs before the trajectory is completed. To address this, during calibration instead choose t to be the current time and t_0 some prior time. Then when "predicting" $\mathbf{x}(t)$ use the logged (vs. planned) sequence of inputs and withhold any state measurements that occurred during the interval.

Systematic prediction. The prediction is the expected value of the measurement, or in this case the residual $\mathbf{r}$. If the parameters $\mathbf{p}$ are correctly identified, then the expected value of the residual is zero:

$$\mathbf{h}_\mu(\mathbf{p}) = \mathrm{Exp}[\mathbf{z}_\mu] = \mathrm{Exp}[\mathbf{r}(t)] = \mathbf{0} \tag{3.8}$$

Though the expected value is zero, residuals will likely never be zero in practice due to noise in the initial/final measurements of state, and in the process.

Systematic measurement covariance. We compute the systematic error covariance via linearized error dynamics, as explained in Stengel's book *Optimal Estimation and Control* [122]. The systematic linearized error dynamics are:

$$\delta\dot{\mathbf{x}}(t) = F(t)\delta\mathbf{x}(t) + G(t)\delta\mathbf{u}(t) \tag{3.9}$$

Where F and G are partial derivatives of the function $\mathbf{f}()$ in the system differential equation (3.1):

$$F = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}, \quad G = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \tag{3.10}$$

We add a random vector $\mathbf{w}$ to the input error to account for process noise. We assume white noise sampled from a zero-mean Gaussian probability density function.

$$\delta\dot{\mathbf{x}}(t) = F(t)\delta\mathbf{x}(t) + G(t)\left(\delta\mathbf{u}(t) + \mathbf{w}(t)\right) \tag{3.11}$$

This is nearly equivalent to equation 4.2-35 in [122]:

$$\dot{\mathbf{x}}(t) = F(t)\mathbf{x}(t) + G(t)\mathbf{u}(t) + L(t)\mathbf{w}(t) \tag{3.12}$$

except that instead of having a linear system, we have locally linearized a nonlinear system, and assumed $L = G$.

Because $\delta\dot{\mathbf{x}}(t)$ is a function of a random vector, it too is a random vector. By squaring and taking the expectation of the systematic error dynamics in (3.11), we obtain the stochastic error dynamics:

$$\dot{P}(t) = F(t)P(t) + P(t)F(t)^T + G(t)Q'(t)G(t)^T \tag{3.13}$$

where:

$$P(t) = \mathrm{Exp}\left[\delta\mathbf{x}(t)\delta\mathbf{x}(t)^T\right] \tag{3.14}$$

P is the second moment (or covariance) of the state error, and Q' is the spectral density of the input noise ($\mathbf{w}$). (3.13) is equivalent to equation 4.2-79 in [122], again assuming that $L = G$. By integrating (3.13) we can propagate uncertainty from time t_0 to t for our state prediction:

$$P(t) = P(t_0) + \int_{t_0}^{t} \left(F(\tau)P(\tau) + P(\tau)F(\tau)^T + G(\tau)Q'(\tau)G(\tau)^T\right) d\tau \tag{3.15}$$

Note that, like $\mathbf{x}_{pred}(t)$ in (3.4), $P(t)$ in (3.15) is not merely a function but a functional, which maps the values of the function $Q'(\tau)$ over an entire interval to another value.

We now derive the measurement covariance using the definition of $\mathbf{z}_\mu$:

$$\begin{aligned} R_\mu &= \mathrm{Cov}[\mathbf{z}_\mu] && (3.16)\\ &= \mathrm{Cov}[\mathbf{r}(t)] && (3.17)\\ &= \mathrm{Cov}[\mathbf{x}_{meas}(t) - \mathbf{x}_{pred}(t)] && (3.18)\\ &= \mathrm{Cov}[\mathbf{x}_{meas}(t)] + \mathrm{Cov}[\mathbf{x}_{pred}(t)] && (3.19) \end{aligned}$$

The covariance of the predicted state is $P(t)$, computed using (3.15):

$$R_\mu = \mathrm{Cov}[\mathbf{x}_{meas}(t)] + P(t) \tag{3.20}$$

Finally, we will call the sensor-dependent covariance of state measurements Σ_{meas}:

$$R_\mu = \Sigma_{meas}(t) + P(t) \tag{3.21}$$

Note that $P(t_0)$ in (3.15) is $\Sigma_{meas}(t_0)$. Note also that even if the initial state at time t_0 is measured perfectly, we still do not expect to predict the final state at time t perfectly due to process noise ($\mathbf{w}$).

In practice, a discrete time relation is used to propagate uncertainty:

$$\begin{aligned} P_k &= \Phi_{k-1}P_{k-1}\Phi_{k-1}^T + Q_{k-1} && (3.22)\\ \Phi_{k-1} &= I + F_{k-1}\Delta t && (3.23)\\ Q_{k-1} &= G(t_k)Q'(t_k)G(t_k)^T\Delta t && (3.24) \end{aligned}$$

Here, the k and $k-1$ subscripts denote time steps, i.e. P_k denotes $P(t_k)$. Φ is called the "state transition matrix." This is the discrete time form of (3.13). The derivation of (3.22) can be found in [122] (see equations 4.2-51, 4.2-53, 4.2-70).

Systematic measurement Jacobian. The Jacobian of the systematic measurement with respect to parameters:

$$H_\mu = \frac{\partial \mathbf{h}_\mu(\mathbf{p})}{\partial \mathbf{p}} \tag{3.25}$$

can be computed via finite differences. We will use the notation $\mathbf{u}(*)$ to mean the entire sequence of inputs between t_0 and t. Then we can suppress the integral notation in (3.4):

$$\mathbf{x}_{pred}(t) = \mathbf{g}\left(\mathbf{x}_{meas}(t_0), \mathbf{u}(*), \mathbf{p}\right) \tag{3.26}$$

This notation makes the functional nature of $\mathbf{x}_{pred}(t)$ clear.

The partial derivative with respect to the i^{th} parameter may be computed via the forward difference:

$$\frac{\partial \mathbf{h}_\mu}{\partial p_i} = -\frac{\partial \mathbf{g}}{\partial p_i} \approx -\frac{\mathbf{g}\left(\mathbf{x}_{meas}(t_0), \mathbf{u}(*), \mathbf{p} + \boldsymbol{\delta}_i\right) - \mathbf{g}\left(\mathbf{x}_{meas}(t_0), \mathbf{u}(*), \mathbf{p}\right)}{\epsilon} \tag{3.27}$$

where $\boldsymbol{\delta}_i$ is a vector of zeros, except for an epsilon (ϵ) at the i^{th} index. The vectors for partial derivatives with respect to each systematic parameter can be concatenated as columns in the matrix H_μ.

The computation time required to compute H_μ using finite differences scales linearly in the number of parameters. Computation should be tractable if forward simulation of the dynamics (i.e. evaluation of $\mathbf{g}$) is much faster than real-time. [108] explains how H_μ can be computed more efficiently under certain conditions: when the system differential equation (3.1) and its Jacobian with respect to the parameters are known analytically, and when the parameters are restricted to modulating the inputs.

3.2.2 Stochastic Variables

Here we define variables for stochastic model calibration (or stochastic calibration for short). By "stochastic model" we refer to the linearized error dynamics for nonsystematic error expressed in (3.13) and (3.15). Calibration of the spectral density Q' enables the characterization of nonsystematic error, and thus the quantification of uncertainty.

The usual approach to quantifying uncertainty is to repeat the same process (or trajectory) numerous times, then compute a sample covariance from the outcomes. But, this characterizes uncertainty for just one trajectory, which may not generalize to other trajectories. Furthermore, we desire to calibrate during normal operation in which trajectories may not be repeated.

Calibration to non-repeated trajectories is possible, because our chosen stochastic error dynamics (3.15) are trajectory dependent in a fortuitous way. Any trajectory provides information about the spectral density Q', and by calibrating Q' we can compute the covariance of any trajectory.

The sensor noise covariance (Σ_{meas} in (3.21)) can also be calibrated using IPEM, but it can often be obtained from the technical specifications of the sensor instead. Calibrating Q' is more critical, because it determines trajectory dependence.

Our stochastic calibration method makes some assumptions. First, we assume that the mean error of the residuals is zero, which is reasonable once the systematic parameters ($\mathbf{p}$) are well calibrated. If systematic and stochastic parameters are calibrated simultaneously, initial estimates of Q' will be overly large until the systematic calibration converges.

We also assume linear dynamics for nonsystematic error. The true stochastic process may be nonlinear, but a linearized approximation is close enough for many applications, and is computationally tractable. This same assumption is made in Kalman filter-based state estimation.

Stochastic measurement. We define the stochastic measurement to be:

$$\mathbf{z}_\sigma = \mathrm{vec}\big(\mathbf{r}(t)\mathbf{r}(t)^T\big) \tag{3.28}$$

where $\mathbf{r}(t)$ is the residual from (3.3). The stochastic measurement ($\mathbf{z}_\sigma$) is just the outer product of the systematic measurement ($\mathbf{z}_\mu$). $\mathbf{z}_\sigma$ can be considered a scatter matrix, but because trajectories are generally not repeated, it is a scatter matrix for a sample size of one.

The function $\mathrm{vec}()$ reshapes the unique elements of a symmetric matrix into a vector. Specifically, the vector's elements are taken from the upper triangular part of the matrix in row-major order. For example, for a three element residual:

$$\mathbf{r}(t) = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \tag{3.29}$$

$$\mathbf{z}_\sigma(t) = \mathrm{vec}\left(\begin{bmatrix} r_1 r_1 & r_1 r_2 & r_1 r_3 \\ r_2 r_1 & r_2 r_2 & r_2 r_3 \\ r_3 r_1 & r_3 r_2 & r_3 r_3 \end{bmatrix}\right) = \begin{bmatrix} r_1 r_1 \\ r_1 r_2 \\ r_1 r_3 \\ r_2 r_2 \\ r_2 r_3 \\ r_3 r_3 \end{bmatrix} \tag{3.30}$$

Stochastic prediction. The stochastic measurement is a noisy observation of the residual covariance. We already derived an equation for the residual covariance using linearized error dynamics in Section 3.2.1.

$$\mathbf{h}_\sigma(\mathbf{s}) = \mathrm{vec}(\mathrm{Cov}[\mathbf{r}(t)]) = \mathrm{vec}(R_\mu) \tag{3.31}$$

where R_μ is defined in equation (3.21). $\mathbf{s}$ denotes a vector of stochastic parameters from which Q' can be computed.

Stochastic measurement covariance. The expected value and covariance of the residual $\mathbf{r}(t)$ are known to be:

$$\mathrm{Exp}[\mathbf{r}(t)] = \mathbf{0} \tag{3.32}$$

$$\mathrm{Cov}[\mathbf{r}(t)] = R_\mu \tag{3.33}$$

To compute the stochastic measurement covariance (R_σ) we must derive the covariance of the residual's *outer product*. Computing the variance of a sample variance is common, but this exercise is a more challenging multivariate analogue: computing the covariance of a sample covariance.

Here is an example for the three element residual from (3.29):

$$R_\sigma = \begin{bmatrix} \mathrm{C}[r_1r_1, r_1r_1] & \mathrm{C}[r_1r_1, r_1r_2] & \mathrm{C}[r_1r_1, r_1r_3] & \mathrm{C}[r_1r_1, r_2r_2] & \mathrm{C}[r_1r_1, r_2r_3] & \mathrm{C}[r_1r_1, r_3r_3] \\ & \mathrm{C}[r_1r_2, r_1r_2] & \mathrm{C}[r_1r_2, r_1r_3] & \mathrm{C}[r_1r_2, r_2r_2] & \mathrm{C}[r_1r_2, r_2r_3] & \mathrm{C}[r_1r_2, r_3r_3] \\ & & \mathrm{C}[r_1r_3, r_1r_3] & \mathrm{C}[r_1r_3, r_2r_2] & \mathrm{C}[r_1r_3, r_2r_3] & \mathrm{C}[r_1r_3, r_3r_3] \\ & & & \mathrm{C}[r_2r_2, r_2r_2] & \mathrm{C}[r_2r_2, r_2r_3] & \mathrm{C}[r_2r_2, r_3r_3] \\ & & & & \mathrm{C}[r_2r_3, r_2r_3] & \mathrm{C}[r_2r_3, r_3r_3] \\ & & & & & \mathrm{C}[r_3r_3, r_3r_3] \end{bmatrix} \tag{3.34}$$

where $\mathrm{C}[\ ]$ is an abbreviation for $\mathrm{Cov}[\ ]$. Only the upper triangular part is shown, but note that R_σ is symmetric.

The elements of this matrix are covariances of *products* of random variables; they can be computed using this rule from Bohrnstedt and Goldberger [15]:

$$\mathrm{C}[ab, cd] = \mathrm{C}[a, c]\,\mathrm{C}[b, d] + \mathrm{C}[a, d]\,\mathrm{C}[b, c] \tag{3.35}$$

The covariances of *single* random variables (or elements of the residual) are obtained from R_μ. Note that the same random variable can be represented by multiple dummy variables in (3.35). For example if $a = c = r_1$, then $\mathrm{C}[a, c] = \mathrm{C}[r_1, r_1]$, which is just the variance of r_1 or element (1,1) of R_μ. Note also that this rule requires that all variables (a,b,c,d) have an expected value of zero.

In general, the equation for element (i, j) of R_σ is:

$$R_\sigma(i, j) = R_\mu(I(i), I(j))R_\mu(J(i), J(j)) + R_\mu(I(i), J(j))R_\mu(J(i), I(j)) \tag{3.36}$$

where I is a vector of row indices, and J is a vector of column indices, obtained as follows:

$$I = \mathrm{vec}\left(\begin{bmatrix}1\\ \vdots \\ m\end{bmatrix}\mathbf{1}^T\right) \tag{3.37}$$

$$J = \mathrm{vec}\big(\mathbf{1}\begin{bmatrix}1 & \dots & m\end{bmatrix}\big) \tag{3.38}$$

where $\mathbf{1}$ is a $m \times 1$ vector of ones.

Stochastic measurement Jacobian. The spectral density matrix $Q'(\tau)$ in (3.15) is always symmetric, such that it can be specified by just its upper triangular part. In the simplest case, it is also time-invariant (Q'). Then the vector of stochastic parameters to be identified is simply:

$$\mathbf{s} = \mathrm{vec}(Q') \tag{3.39}$$

Other parametrizations are possible. To ensure that Q' is positive definite one can instead identify its Cholesky decomposition:

$$Q' = U^T U \tag{3.40}$$

$$\mathbf{s} = \mathrm{vec}(U) \tag{3.41}$$

where U is an upper triangular matrix. The process noise may be time-variant or heteroscedastic, but for WMRs, we found a parsimonious parametrization of Q' to be sufficient (Section 3.4.1).

The stochastic measurement Jacobian:

$$H_\sigma = \frac{\partial \mathbf{h}_\sigma(\mathbf{s})}{\partial \mathbf{s}} \tag{3.42}$$

can be computed using finite differences, just like the systematic measurement Jacobian H_μ in (3.27). Also just like H_μ, under certain conditions H_σ can be computed more efficiently as explained in [108].

3.2.3 Systematic vs. Stochastic Comparison

Table 3.2 summarizes the variables for both systematic and stochastic calibration. Notice how the variables interrelate. In particular, the stochastic prediction ($\mathbf{h}_\sigma$) is equal to the systematic measurement covariance (R_μ). This is a unified approach to calibrating both models simultaneously. Note that computing $\mathbf{r}(t)$ requires state measurements ($\mathbf{x}_{meas}$) at the initial time t_0 and final time t. Likewise computing R_μ requires the sensor-dependent covariance of these measurements (Σ_{meas}) at both times.

Table 3.2: Systematic and stochastic calibration variables

	Systematic (μ)	Stochastic (σ)
parameters	$\mathbf{p}$	$\mathbf{s}$
measurement ($\mathbf{z}$)	$\mathbf{r}(t)$	$\mathbf{r}(t)\mathbf{r}(t)^T$
prediction ($\mathbf{h}$)	$\mathbf{0}$	$\Sigma_{meas}(t) + P(t)$
meas. covariance (R)	$\Sigma_{meas}(t) + P(t)$	$\mathrm{Cov}\left[\mathbf{r}(t)\mathbf{r}(t)^T\right]$

3.2.4 Offline and Online Algorithms

IPEM can be implemented in both offline and online algorithms. Offline batch algorithms are suitable when conditions are fixed. Online algorithms allow adaptation to changing conditions, such as hardware damage, payload shifts, or terrain transitions.

Offline. In offline calibration, we solve for optimal parameter values ($\mathbf{p}^*$, $\mathbf{s}^*$) that minimize explicit objective functions. To calibrate the systematic model, one could simply minimize residuals in a least-squares sense:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \sum_{k=1}^{n} \mathbf{r}_k^T \mathbf{r}_k \tag{3.43}$$

where $\mathbf{r}_k$ is the residual for the k^{th} of n trajectories in the data set. But, this does not account for the unique covariance of each residual. Covariance increases with duration ($t - t_0$), so larger residuals are expected for longer trajectories. Accordingly, one could minimize the Mahalanobis distance of residuals:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \sum_{k=1}^{n} \mathbf{r}_k^T (R_\mu)_k^{-1} \mathbf{r}_k \tag{3.44}$$

Below we present an *online*, Kalman filter-based algorithm that minimizes this same objective function.

For stochastic calibration, the corresponding optimization to (3.44) would be:

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} \sum_{k=1}^{n} (\mathbf{z}_\sigma)_k^T (R_\sigma)_k^{-1} (\mathbf{z}_\sigma)_k \tag{3.45}$$

Abbeel et al. [1] propose maximizing a prediction log likelihood objective. An analagous metric for IPEM would be:

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} \sum_{k=1}^{n} \log\left(p(\mathbf{r}_k)\right) \tag{3.46}$$

$$p(\mathbf{r}_k) = \mathcal{N}\left(\mathbf{r}_k; \mathbf{0}, (R_\mu)_k\right) \tag{3.47}$$

where $\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \Sigma)$ denotes the formula for a multivariate Gaussian probability density function with mean $\boldsymbol{\mu}$ and covariance Σ, evaluated at $\mathbf{z}$.

The objective functions above can be minimized by a variety algorithms. Nelder-Mead, or the downhill simplex method, is a popular derivative-free option. Abbeel et al. used a simpler coordinate ascent algorithm.

Online. Here we present an online implementation of the IPEM method based on the extended Kalman filter (EKF). The EKF makes utilizing measurement covariances (R) straightforward, and provides explicit control over the weight of history vs. the present measurement via the Q matrix.

The process update equations for the identification EKF are:

$$\boldsymbol{\xi}_{k|k-1} = \boldsymbol{\xi}_{k-1|k-1} \tag{3.48}$$

$$\mathcal{P}_{k|k-1} = \mathcal{P}_{k-1|k-1} + Q_k \tag{3.49}$$

$\boldsymbol{\xi}$, $\mathcal{P}$, Q in the identification EKF are not to be confused with analagous variables in the system dynamics ($\mathbf{x}$, P, Q). $\boldsymbol{\xi}$ here is the vector of parameters to be identified, composed of $\mathbf{p}$ and/or $\mathbf{s}$. This process update leaves the parameter estimates unchanged while increasing their covariance $\mathcal{P}$ by Q. The larger Q is, the more weight is placed on the present measurement.

The measurement update equations for the identification EKF are:

$$\mathbf{y}_k = \mathbf{z}_k - \mathbf{h}(\boldsymbol{\xi}_{k|k-1}) \tag{3.50}$$

$$S_k = H_k \mathcal{P}_{k|k-1} H_k^T + R_k \tag{3.51}$$

$$K_k = \mathcal{P}_{k|k-1} H_k^T S_k^{-1} \tag{3.52}$$

$$\boldsymbol{\xi}_{k|k} = \boldsymbol{\xi}_{k|k-1} + K_k \mathbf{y}_k \tag{3.53}$$

$$\mathcal{P}_{k|k} = (I - K_k H_k)\mathcal{P}_{k|k-1} \tag{3.54}$$

where $\mathbf{z}$, $\mathbf{h}$, R, and H are composed of the systematic and/or stochastic variables defined in Sections 3.2.1 and 3.2.2. S is the covariance of the innovation ($\mathbf{y}$), and K is the Kalman gain.

Note that the stochastic calibration (i.e. the calibration of the spectral density Q' through $\mathbf{s}$) affects the measurement covariance in the systematic calibration (R_μ). Sometimes large residuals due to poor initial estimates of the systematic parameters ($\mathbf{p}$) cause overestimation of Q', which in turn slows the rate of convergence. This can be addressed by bounding Q', at least initially.

3.3 Application of IPEM to WMR Models

This section applies IPEM to calibration of the kinematic/dynamic WMR model formulations presented in Chapter 2.

3.3.1 Systematic WMR Model Calibration

Here we map terms in the system differential equation (3.1) to our WMR model formulations. Systematic calibration removes systematic errors from WMR motion predictions.

For kinematic models (Section 2.2), the state $\mathbf{x}$ is simply the state $\mathbf{q}$ defined in (2.10), and the input vector $\mathbf{u}$ is the joint space velocity $\dot{\mathbf{q}}'$. Previously in (2.73) we used $\mathbf{u}$ to represent a vector of commanded joint rates, but this is not to be confused with its usage here. The system differential equation is:

$$\dot{\mathbf{q}}(t) = V(\mathbf{q}(t))\dot{\mathbf{q}}'(t) \tag{3.55}$$

where the matrix V is defined in (2.14). The joint space velocity $\dot{\mathbf{q}}'$ is computed via our stabilized DAE method (Section 2.2.5). It is a function of the WMR state, known joint rates, the terrain geometry, and parameters ($\mathbf{p}$). These parameters can be dimensions in the frame information, or coefficients in the body-level slip parametrization (Section 2.3).

Per (3.4), to compute the residual we integrate the system dynamics (or forward simulate the kinematic model) from time t_0 to t. The residual is nominally:

$$\mathbf{r}(t) = \mathbf{q}_{meas}(t) - \mathbf{q}_{pred}(t) \tag{3.56}$$

however, the entire state vector ($\mathbf{q}$) is often not observable, nor does it need to be. For example, the residual might only comprise errors in the predicted (x,y) location and yaw of the body frame. Roll and pitch measurements can optionally be treated as inputs in the simulation.

$\mathbf{q}$ contains the pose of the vehicle body frame, but pose measurements are typically of a mounted sensor that is not collocated with body frame. We can account for this offset as follows, expressing poses as homogeneous transforms. At the start of the interval, we obtain the initial body pose with respect to the world (T_b^w) from the measured sensor pose (T_s^w) and the known sensor offset (T_s^b):

$$T_b^w(t_0) = (T_s^b)^{-1} T_s^w(t_0) \tag{3.57}$$

We can then simulate over the interval as usual, to predict the final pose of the body frame. Afterward, we obtain the predicted the final pose of the sensor as follows:

$$T_s^w(t) = T_s^b T_b^w(t) \tag{3.58}$$

The residual $\mathbf{r}(t)$ is the difference between measured and predicted *sensor* pose at the final time t. The sensor pose offsets can even be calibrated via IPEM simultaneously with other parameters, as we demonstrate in Section 3.4.2.

For dynamic models (Section 2.4), the system differential equation is:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{q}(t)\\ \dot{\mathbf{q}}'(t)\end{bmatrix} = \begin{bmatrix}V(\mathbf{q}(t)) & \\ & I\end{bmatrix}\begin{bmatrix}\dot{\mathbf{q}}'(t)\\ \ddot{\mathbf{q}}'(t)\end{bmatrix} \tag{3.59}$$

The joint space acceleration $\ddot{\mathbf{q}}'(t)$ is computed via our DAE/force-balance optimization method (Section 2.4.3) and integrated twice. For dynamic models, the parameters $\mathbf{p}$ can be dimensions, mass properties, and constants in the wheel-ground contact and actuator models.

3.3.2 Stochastic WMR Model Calibration

Here we map terms in the stochastic error dynamics (3.13) to our WMR model formulations. Stochastic calibration characterizes the stochasticity of WMR motion due to random disturbances, such that the uncertainty of motion predictions can be quantified.

For kinematic models, the Jacobian matrix F is defined as follows:

$$F = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \frac{\partial V(\mathbf{q})\dot{\mathbf{q}}'}{\partial \mathbf{q}} \tag{3.60}$$

Based on the definition of V in (2.14), F can be considered a *block* 3×3 matrix with only two nonzero blocks:

$$F([1],[1]) = \frac{\partial \Omega \vec{\omega}_b^w}{\partial \mathbf{o}_b^w} \tag{3.61}$$

$$F([2],[1]) = \frac{\partial R_b^w \vec{v}_b^w}{\partial \mathbf{o}_b^w} \tag{3.62}$$

With some manipulation, it can be shown that: $F([2],[1]) = -R_b^w[\vec{v}_b^w]_\times \Omega^{-1}$

Finally, the Jacobian matrix G is simply:

$$G = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = V(\mathbf{q}) \tag{3.63}$$

3.4 Experimental Evaluation

This section assesses the accuracy of WMR models relative to data collected in physical experiments. These models were formulated as explained in Chapter 2 and calibrated using IPEM as explained in this chapter. First, Section 3.4.1 presents the online calibration of an enhanced kinematic model of Crusher. Section 3.4.2 compares the accuracy of kinematic and dynamic models for multiple vehicles and terrain types. Finally, Section 3.4.3 presents 3D odometry results for the Zoë rover while traversing obstacles.

Our motion models are relevant to both estimation and prediction tasks. In estimation, inputs are sensed values which usually include attitude, wheel velocities, and joint angles (for steering, suspension). In prediction (such as for model predictive planning/control), inputs are planned commands and terrain geometry generated via perception. Attitude is determined by the enforcement of holonomic contact constraints with respect to the terrain geometry.

In our experiments, "prediction" results were generated offline by reading values from data logs rather than receiving them directly from the planner and sensors. Terrain geometry perception was not available for most logs, so IMU attitude measurements were used as inputs. Nevertheless, we claim that these results are indicative of predictive performance, provided a good actuator model (which maps commands to actual joint rates/angles) and a good perception system. Successful actuator model calibration is demonstrated in [107]. In some prior experiments on Crusher for which laser scanner data was logged, we found little difference in predictive performance when using terrain geometry vs. IMU attitude measurements.

3.4.1 Online Calibration Results

This section presents the online calibration of a Crusher vehicle model to data collected at Camp Roberts, CA. We have presented results for this data log in prior work [108], but here we use the enhanced kinematic model formulation presented in Section 2.3. In contrast to the formulation used in [108], the new formulation accounts for suspension deflections and parametrizes body-level slip in a more principled way.

NREC's Crusher is a six-wheel skid-steer vehicle designed for extreme mobility (Figure 3.2). An advanced suspension with independent linkages for each wheel enables Crusher to climb over obstacles and very rough terrain.

In this experiment, Crusher traversed steep grassy slopes and a dirt road at Camp Roberts. Crusher's path is plotted in Figure 3.3. During the experiment, elevation varied by 40 m, roll angles varied from approximately -30 to 30°, and pitch angles varied from -20 to 20°. As shown in Figure 3.4, Crusher reached speeds up to 6 m/s and 40 deg/s.

Ground truth was obtained via Real Time Kinematic (RTK) GPS and an inertial navigation system (INS), but this is not required in general. IPEM could suffice with only relative pose measurements via visual odometry, for example. Because predictions are integrated, ground truth can even be obtained by driving between surveyed waypoints.

Figure 3.2: A Photograph of the Crusher vehicle, built by the National Robotics Engineering Center (NREC) with support from DARPA.

In this experiment we calibrated just six parameters, i.e. the six coefficients in the parametrization of body-level slip in the enhanced kinematic model (see (2.48) and (2.49) in Section 2.3). These were calibrated using the online EKF-based IPEM algorithm presented in Section 3.2.4. The procedure was as follows:

Data from a 360 s log was passed to the identification algorithm chronologically, just as it was received, in non-overlapping 2 s intervals. For each interval, vehicle pose is predicted at the final time (t), based on the GPS/INS pose measurement at the initial time (t_0), and some measurements logged during the interval (wheel velocities, suspension angles, roll/pitch angles). The prediction does *not* use any measurements of position or yaw logged during the interval. Slip is predicted using the current estimates of the parameter values, based on calibration to *prior intervals*. A residual is formed by differencing the measured and predicted final pose, per (3.56). Based on this residual, the parameter estimates are updated per (3.53) and the process is repeated for the next interval.

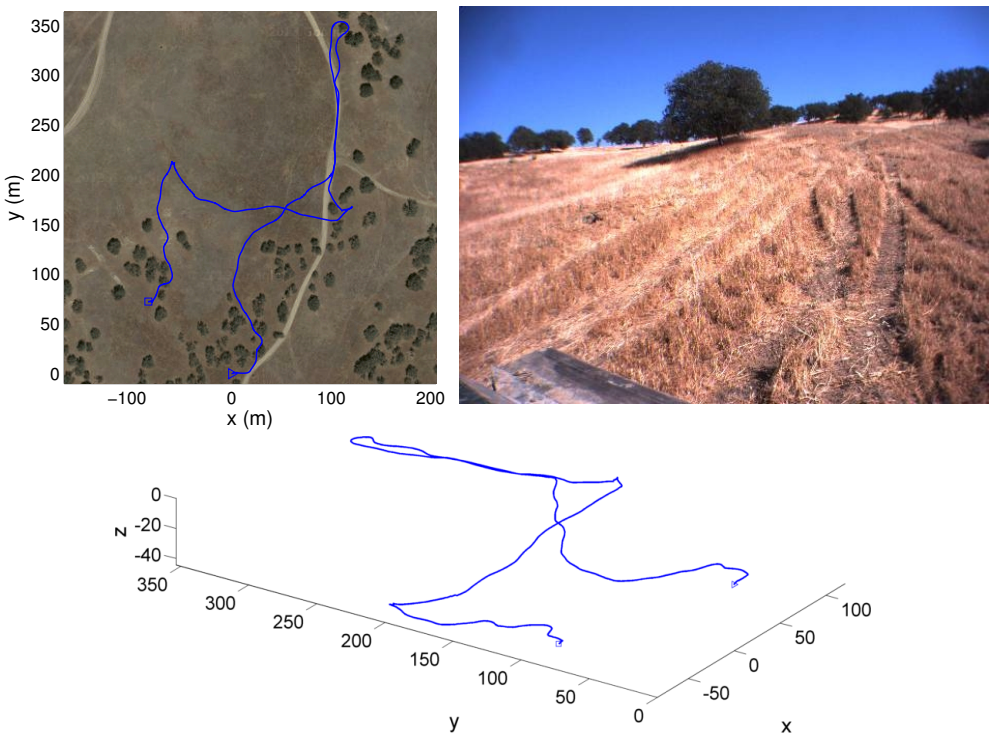


Figure 3.3: (Top Left) Path of Crusher at Camp Roberts measured via GPS, plotted on an aerial image. (Top Right) An image captured by one of Crusher's cameras during the experiment. (Bottom) A 3D plot of Crusher's path. Crusher traversed steep grassy slopes and a dirt road.

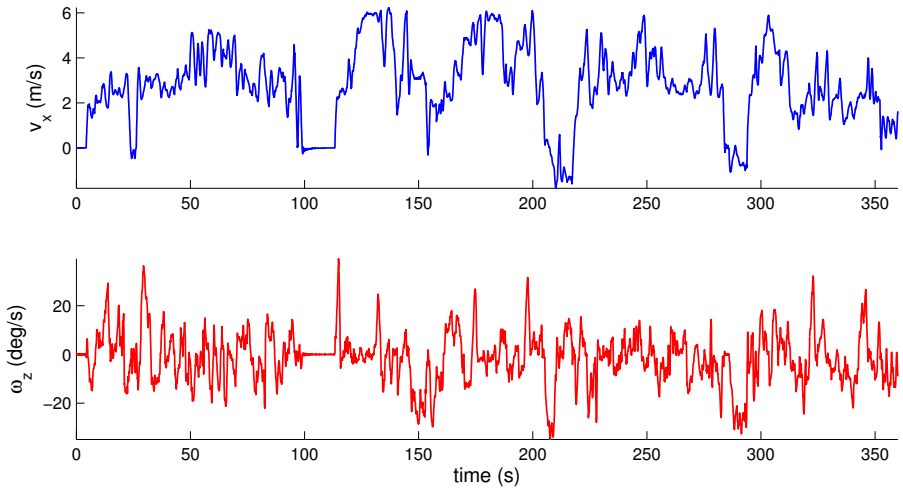


Figure 3.4: Linear and angular velocity vs. time for Crusher at Camp Roberts, measured via GPS/INS. Linear velocity is measured about the body frame x (or forward axis), angular velocity is measured about the z (or vertical) axis.

Other interval lengths (1, 4 s) were tested, but for brevity are not presented here. When choosing a horizon for MPP, one should consider the time required to come to an emergency stop, the range of the perception sensors, and uncertainty.

Systematic Calibration Results.

We assessed the predictive accuracy of four Crusher vehicle models:

- *diff. drive*: differential drive. The vehicle moves instantaneously in a plane tangent to the terrain surface. The vehicle's forward velocity along x (v_x) and angular velocity about z (ω_z) are defined as follows:

$$v_x = (v_l + v_r)/2 \tag{3.64}$$
$$\omega_z = (v_r - v_l)/B \tag{3.65}$$

  Where B is the track width, and the wheel velocities v_l and v_r are products of wheel radius and angular velocity.

- *diff. drive+*: enhanced differential drive model. This model is equivalent to the diff. drive model, except that angular velocity is scaled linearly:

$$\omega_z = \alpha(v_r - v_l)/B \tag{3.66}$$

- *3D kin.*: 3D kinematic model (Section 2.2). Frame information for Crusher is provided in Table E.3 in Appendix E.
- *3D kin.+*: enhanced 3D kinematic model (Section 2.3)

Table 3.3 presents errors in predicted final position and yaw for all models, averaged over all 2 s intervals. The "online" results for the enhanced models (rows 2 and 5) represent prediction errors during online calibration; each interval was predicted using parameter estimates based on calibration only to *prior intervals*. The "converged" results represent prediction errors using the final parameter estimates after calibrating to all intervals. We calibrated the α parameter in the diff. drive+ model via the same online IPEM algorithm used for the 3D kin.+ model.

Table 3.3: Pose prediction errors for Crusher at Camp Roberts

model	mean pos. (m)[1]	mean yaw (rad)[2]
diff. drive	1.2563	0.3803
diff. drive+ online	0.7319	0.1007
diff. drive+ converged	0.6989	0.0797
3D kin.	0.6603	0.0731
3D kin.+ online	0.5583	0.0743
3D kin.+ converged	0.5402	0.0715

errors are averages over all 2 s intervals
1. Euclidean distance error of final predicted position
2. absolute error of final predicted yaw

The non-enhanced differential drive model greatly overestimates the yaw rate, causing poor predictive accuracy. Adding the α parameter to scale the yaw rate reduces error significantly. This diff. drive+ model is a common approximation used for skid-steer vehicles in related work

[82][137]; however, it neglects suspension deflections and wheel slip caused by gravity and acceleration.

Even our non-enhanced 3D kinematic model outperforms the diff. drive+ model. Our enhanced 3D kinematic model performs the best of all by accounting for non-zero slip. When using converged parameter estimates, 3D kin.+ position errors are reduced by 57% relative to the diff. drive model, and 29% relative to the diff. drive+ model. A more dramatic improvement for enhanced vs. non-enhanced 3D kinematic models is observed for the LandTamer and RecBot in Section 3.4.2. Prediction errors for Crusher are much larger than for the LandTamer and RecBot overall, due to high speeds and inhomogeneous terrain.

Encouragingly, the 3D kin.+ *online* results are nearly as good as the *converged* results, suggesting that the model converged quickly. Online calibration is more convenient than offline/batch calibration; it can run during normal operation and provide immediate benefit. In addition, it enables adaptation to changing conditions in real-time.

The estimated parameter values vs. iteration (or interval number) for the 3D kin.+ model are plotted in Figure 3.5. The estimates converge quickly, within about 20 iterations or 40 seconds of driving. Of course, the rate of convergence depends on how well the trajectories cover the state space. If a vehicle never turns, for example, parameters of the model that affect turning behavior cannot be calibrated. Trajectories in this experiment varied sufficiently to cover all inputs in our slip parametrization. In particular, Crusher experienced significant lateral accelerations up to 0.5 g due to gravity and 0.4 g due to high-speed turns.

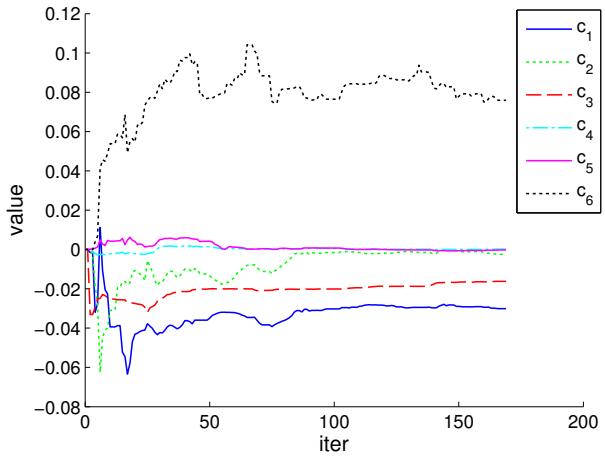


Figure 3.5: Parameter values vs. iteration (interval number) during online calibration of the Crusher vehicle model.

Position prediction errors can be visualized in the scatter plots in Figure 3.6. Each dot represents the position error for a single 2 s interval, during which Crusher might traverse up to 12 m. The enhanced 3D kinematic model greatly reduces error in the cross-track (y) direction relative to the differential drive model, due to better modeling of the yaw dynamics.

Measured and predicted trajectories for two representative 4 s intervals are plotted in Figure 3.7. The ellipses represent uncertainty in the 3D kin.+ model predictions at 1 s intervals. These covariances were computed using calibrated estimates of the stochastic parameters.

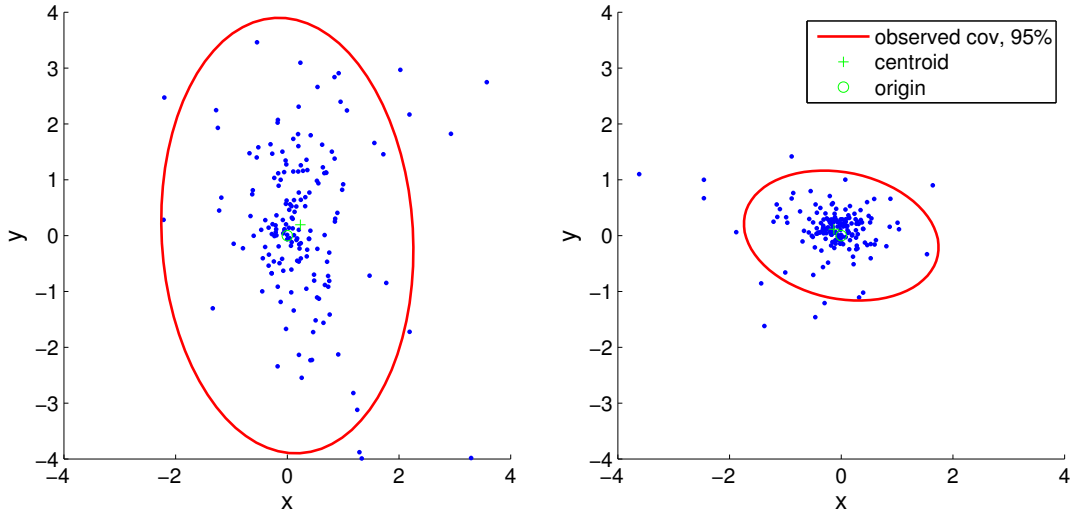


Figure 3.6: Scatter plots of position prediction errors for Crusher at Camp Roberts: (Left) for the differential drive model, (Right) for the enhanced 3D kinematic model after fully converged. Each dot represents the error of final predicted position for a single 2 s interval. Errors are measured along the x and y axes of the *initial* body frame orientation.

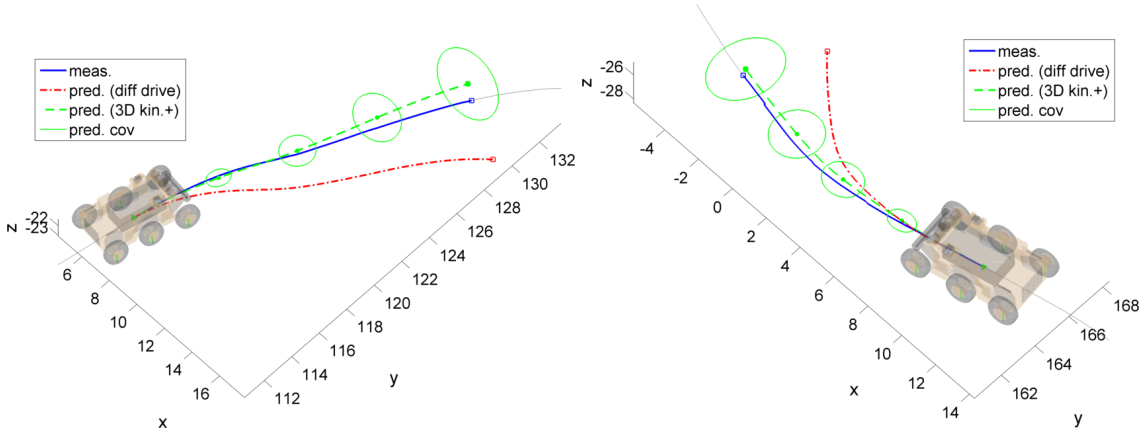


Figure 3.7: Predicted trajectories for two 4 s intervals for Crusher at Camp Roberts. The covariance ellipses represent uncertainty in the 3D kin.+ prediction at 1 s intervals.

Stochastic Calibration Results. Some uncertainty in WMR model predictions is unavoidable due to sensor noise and random disturbances. We can, however, characterize stochastic error propagation in the model to quantify this uncertainty. In this experiment, we calibrated a linearized error dynamics model for stochastic error using our online IPEM algorithm (Sections 3.2.2, 3.2.4). We found the linearized model to adequately fit the data when calibrating to extended intervals.

Figure 3.8 presents scatter plots of position prediction errors just like Figure 3.6, but for intervals of 1, 2, and 4 s. The sample covariances of these errors (for inlier trajectories only) are represented by solid red ellipses. Notice that the covariance grows with time, particularly in the cross-track (y) direction due to the compounded effects of yaw error. Our stochastic model

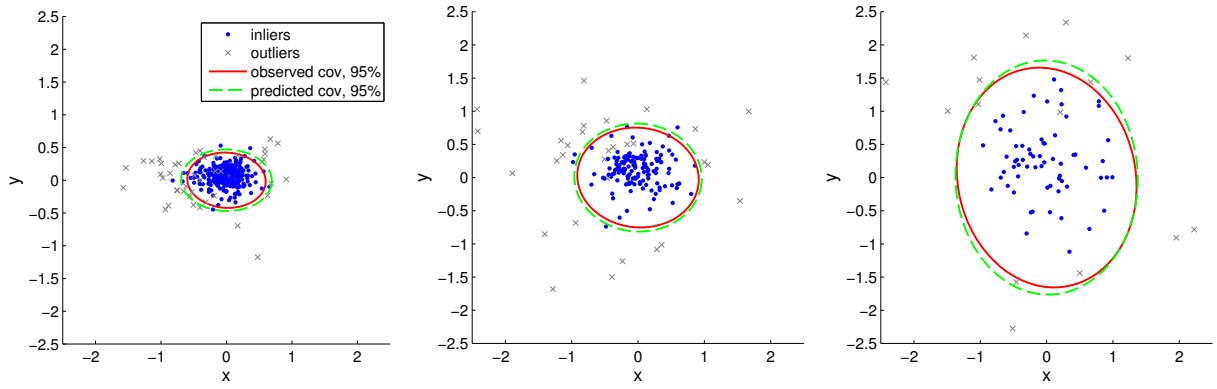


Figure 3.8: Position error scatter plots for 1, 2, and 4 s intervals for Crusher at Camp Roberts. Only the inliers are used to compute covariances. The solid red ellipse in each plot represents the sample covariance computed from the position errors. The dashed green ellipse represents the average of the covariances predicted by the calibrated stochastic model, according to (3.15).

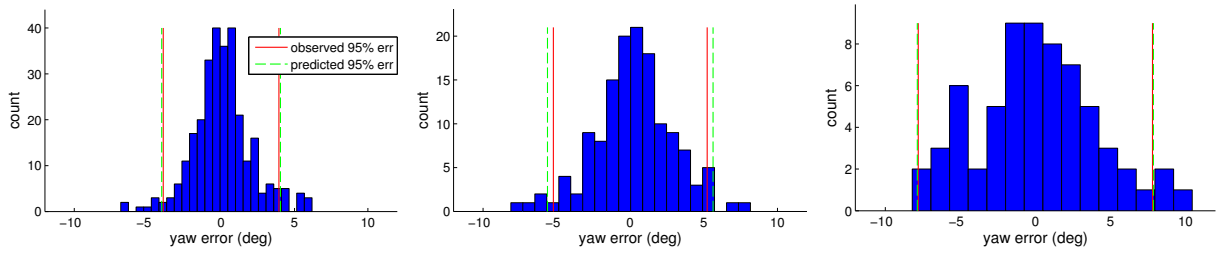


Figure 3.9: Yaw error histograms for 1, 2, and 4 s intervals for Crusher at Camp Roberts (inliers only). The solid red lines represent the sample variance computed from the yaw errors. The dashed green lines represent the average of the yaw variances predicted by the calibrated stochastic model.

predicts a unique, trajectory-dependent covariance for each interval according to (3.15). The average of these covariances is represented by the dashed green ellipses, which closely match the observed scatter. This suggests that our calibrated stochastic model accurately predicts the propagation of uncertainty over time.

Figure 3.9 presents histograms of yaw prediction errors for 1, 2, and 4 s intervals. Just as for the position errors, our calibrated stochastic model accurately predicts the time-dependent growth of yaw error.

We note that our stochastic model fits the data well, despite the violation of some assumptions. First, our model assumes white noise, but the actual noise is significantly time correlated. This is visible in Figure 3.10 which plots both measured and predicted velocity for a 2 s interval; note that errors at neighboring time steps are usually very close. A straightforward approach to calibrate Q' given the whiteness assumption is to compute the sample covariance of the velocity residuals:

$$Q' = \frac{1}{N-1}\sum_{i=1}^{N} \mathbf{r}_i \mathbf{r}_i^T \tag{3.67}$$

$$\mathbf{r} = \dot{\mathbf{q}}'_{meas} - \dot{\mathbf{q}}'_{pred}$$

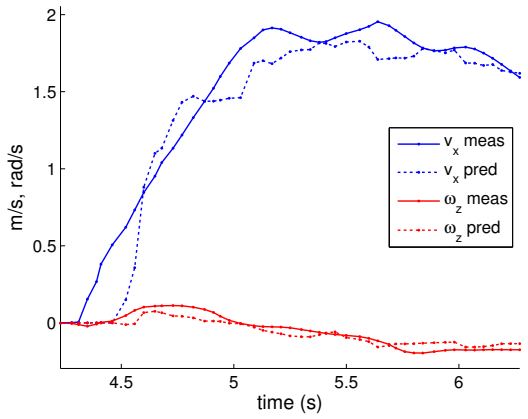
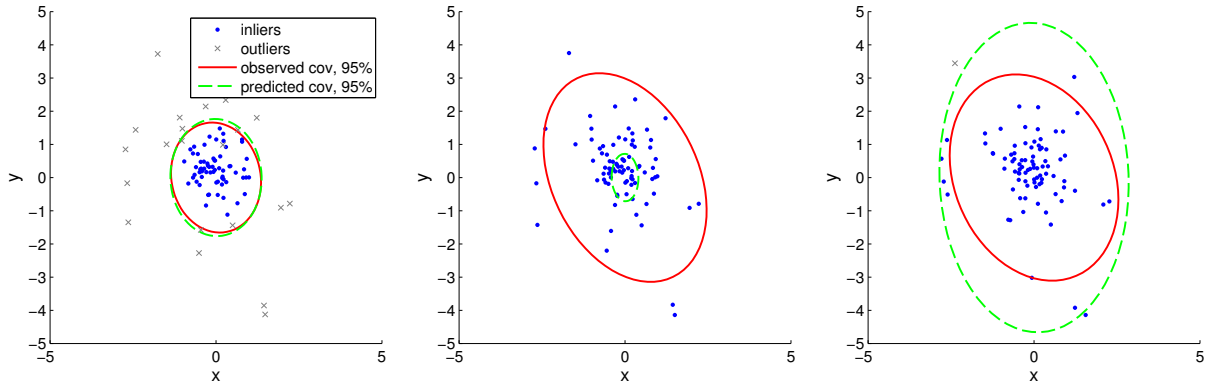


Figure 3.10: Measured vs. predicted forward (v_x) and angular (ω_z) velocity for a 2 s interval of the Crusher at Camp Roberts data log. Velocity residuals are significantly time-correlated.



(a) (b) (c)

Figure 3.11: Position error scatter plots for 4 s intervals for Crusher at Camp Roberts. (a) same as the plot in Figure 3.8 but with wider axes. (b) with stochastic parameters calibrated to velocity residuals. (c) with fewer intervals rejected as outliers. Plots for 1, 2, and 4 s intervals are provided in Appendix F.

If we calibrate Q' this way, it correctly represents the magnitude of instantaneous noise, but P is greatly underestimated for longer intervals (Figure 3.11(b)). When calibrating to pose residuals for longer intervals, Q' is large to compensate for violation of the whiteness assumption, but as a result predictions of P match the data closely.

Second, our model assumes a Gaussian noise distribution, but the actual distribution resembles a Gaussian with an elongated tail due to outliers. Outlier rejection is essential during calibration; without it a single bad trajectory can skew the parameter estimates. Outliers can be identified according to Mahalanobis distance:

$$D_M = \mathbf{y}^T S^{-1} \mathbf{y} \tag{3.68}$$

where $\mathbf{y}$ is the innovation and $\mathbf{S}$ is the innovation covariance from the EKF measurement update (see (3.50) and (3.51)). We chose a high Mahalanobis distance threshold for this experi-

ment (90), but still many trajectories were rejected as outliers (Figure 3.8). This aided convergence of the systematic parameter estimates (i.e. of the body-level slip model) but resulted in a stochastic model that underestimates uncertainty. Sometimes a more conservative uncertainty estimate is desired, such as when assessing risk in motion planning. In that case one can calibrate the stochastic model with minimal or no outlier rejection; the model will not fit the data as closely, due to violation of the Gaussian noise assumption, but it will be more conservative (Figure 3.11(c)).

In conclusion, our 3D kinematic model of the Crusher vehicle predicts motion far more accurately than a differential drive model. Enhancement of the kinematic model to account for nonzero wheel slip further improves accuracy, especially when large gravitational or inertial forces induce slip. Parameters can be calibrated quickly and conveniently using our online IPEM method. Finally, stochastic error can also be accurately characterized using IPEM, to quantify the uncertainty of motion predictions.

3.4.2 Kinematic vs. Dynamic Modeling Results

This section compares the accuracy of kinematic and dynamic model formulations for two vehicles on three terrain types. The objective of these experiments was to analyze the tradeoff between model fidelity and computation speed over a wide range of conditions.

Experiments were conducted on the LandTamer and RecBot vehicles (Figure 3.12). The LandTamer is an amphibious skid-steer vehicle; a dynamic model of the LandTamer was already used in Sections 2.5.2 and 2.5.3. The RecBot is a John Deere Gator that has been retrofitted with sensors/actuators for unmanned operation. The RecBot is Ackerman-steered, meaning that like a car its front wheels are steerable but its rear wheels are not. We omitted the sprung suspension for the RecBot's front wheels from our models, because it is not instrumented and experiments were conducted on relatively smooth terrain. Frame information for both vehicles is provided in Appendix E (Tables E.1 and E.4).

Figure 3.12: (Left) Photograph of the LandTamer, a six-wheeled skid-steer vehicle made by PFM Manufacturing Inc. (Right) Photograph of the RecBot, a John Deere Gator retrofitted for unmanned operation. The RecBot is fully electric and Ackerman-steered.

The paths of the vehicles on asphalt, dirt, and grass are shown in Figures 3.13 and 3.14. As in the Crusher experiment, ground truth was obtained via RTK GPS/INS. The vehicles were driven

randomly at speeds up to 2.9 m/s and 0.8 rad/s for the LandTamer, and 5.3 m/s and 0.9 rad/s for the RecBot.

Models were calibrated to 180 s portions of the data log for each terrain type, using the same online IPEM algorithm used in Section 3.4.1. We assessed the predictive accuracy of three model types:

- *kin.*: 3D kinematic model formulation (Section 2.2). This model assumes no or minimal wheel slip.
- *kin.*+: enhanced 3D kinematic model formulation (Section 2.3). Parameters calibrated include the six coefficients in the body-level slip parametrization.
- *dyn.*: 3D dynamic model formulation (Section 2.4). A piecewise-linear wheel-ground contact model was found to provide the best results (Appendix C). Parameters calibrated include constants in the contact model and (x,y) cg offsets.

For all models (x,y) sensor position offsets were calibrated simultaneously with the other parameters.

LandTamer Results. Prediction errors for the LandTamer models on all terrain types are presented in Table 3.4. The 180 s logs were divided into non-overlapping 10 m segments for calibration; interval durations varied depending on speed. The reported errors are averages over all segments. The "kin.+ online" row represents prediction errors during online calibration, whereas the "kin.+" and "dyn." rows represent errors using fully converged parameter estimates. For convenience, Table 3.5 presents percent reductions in prediction errors relative to the non-enhanced kinematic model (kin.), based on the data in Table 3.4.

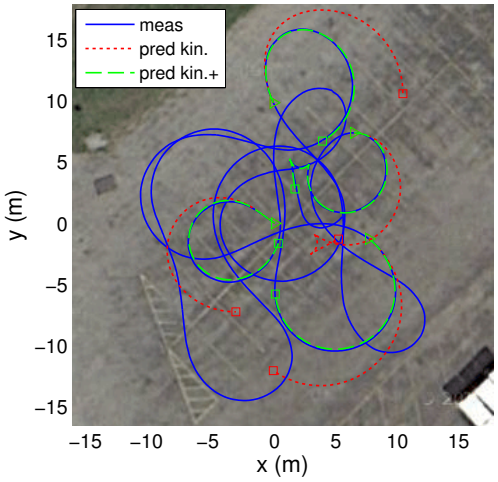
Accounting for nonzero wheel slip, whether through enhanced kinematics or dynamics, significantly improves the accuracy of motion predictions. On pavement, position and yaw prediction errors are reduced by over 90% for a 10 m horizon. Errors are higher for the rougher, less homogeneous dirt and grass terrains, but are still reduced by at least 50% relative to the non-enhanced kinematic model.

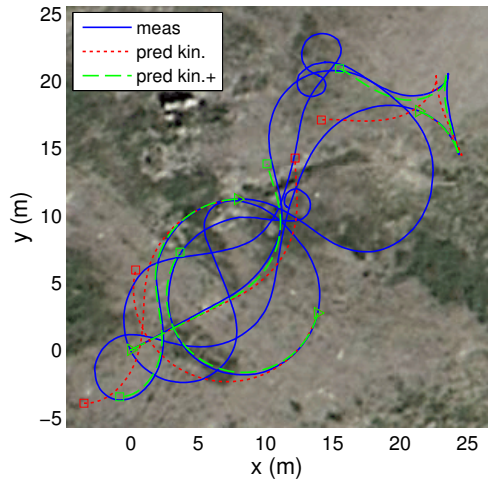
Table 3.4 also presents prediction errors for 20 m segments (using the parameter estimates calibrated to 10 m segments). These indicate that accuracy is improved even for horizons longer than the one calibrated to. The magnitude of these improvements is clearly visible in Figure 3.13, which shows several 20 m predicted path segments using both the non-enhanced and enhanced kinematic models.

Prediction errors during online calibration confirm the Crusher results in Section 3.4.1; online calibration improves accuracy immediately. The observed improvement is not due to overfitting, because predictions are based only on calibration to *prior intervals* (i.e. there is separation of training and test data). The online prediction errors would be closer those using converged parameter estimates if the experiments were longer; here all logs are only 180 s whereas a 360 s log was used in Section 3.4.1.

The predictive accuracies of the enhanced kinematic and dynamic models are comparable in this experiment; however, the enhanced kinematic model is approximately $5\times$ computationally faster. This suggests that, in some cases, using our enhanced kinematic model formulation may be preferable to using a full dynamics formulation.

As explained in Section 3.4.1, for simplicity and speed, skid-steer vehicles are often modeled as differential drive vehicles in related work. Table 3.6 presents prediction errors for differential





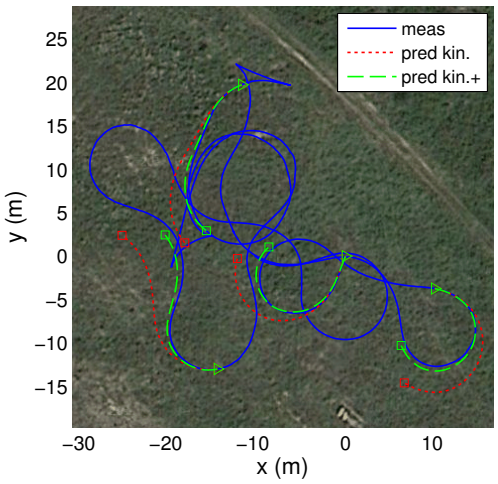
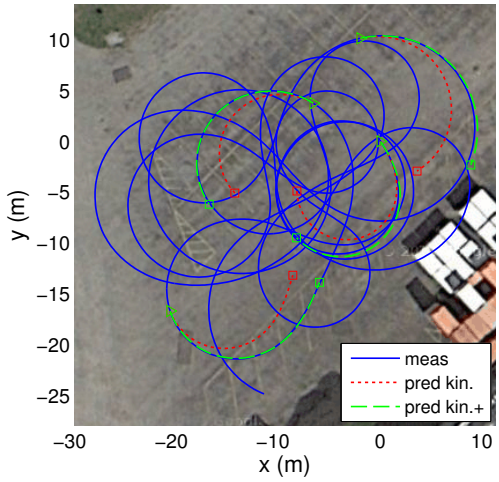
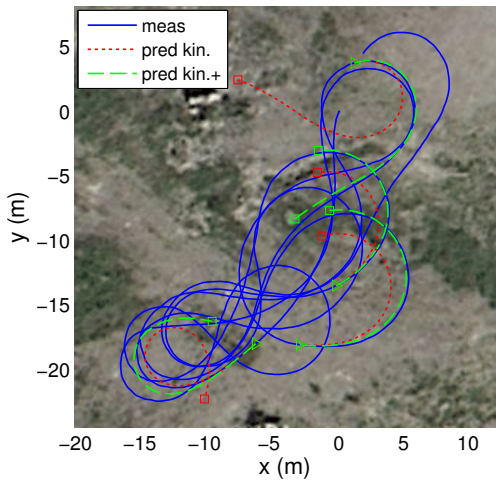


Figure 3.13: Path of the LandTamer on asphalt, dirt, and grass. Predicted paths for several 20 m segments are also shown, using the non-enhanced (kin.) and enhanced (kin.+) kinematic models.





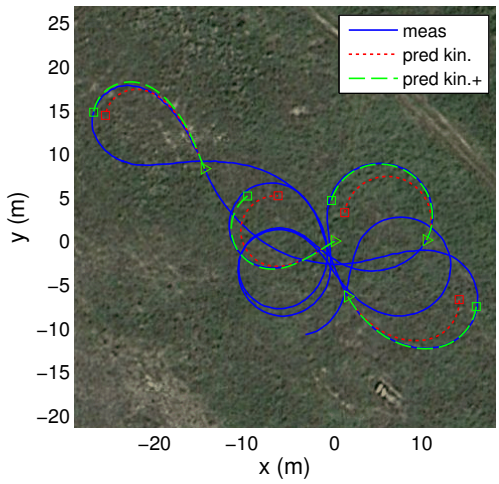


Figure 3.14: Path of the RecBot on asphalt, dirt, and grass.

Table 3.4: Pose prediction errors for the LandTamer on three terrain types

		asphalt		dirt		grass	
horizon	model	pos. err. (m)[1]	yaw err. (rad)[2]	pos. err. (m)	yaw err. (rad)	pos. err. (m)	yaw err. (rad)
10 m	kin.	1.7460	0.4112	1.3957	0.2558	1.2229	0.2148
	kin.+ online	0.2882	0.0528	0.6711	0.1409	0.8295	0.1388
	kin.+	0.1564	0.0131	0.5044	0.1016	0.5956	0.1071
	dyn.	0.1129	0.0241	0.5194	0.0941	0.4513	0.0760
20 m	kin.	4.9431	0.7564	3.9198	0.3503	3.5977	0.3443
	kin.+	0.2135	0.0184	1.3814	0.2095	1.5539	0.1721
	dyn.	0.3085	0.0351	1.3492	0.1682	1.0220	0.1257

errors are averages over all intervals
1. Euclidean distance error of final predicted position
2. absolute error of final predicted yaw

Table 3.5: Percent reductions in LandTamer pose prediction errors[1]

		asphalt		dirt		grass	
horizon	model	pos. err.	yaw err.	pos. err.	yaw err.	pos. err.	yaw err.
10 m	kin.+ online	83.5%	87.2%	51.9%	44.9%	32.2%	35.4%
	kin.+	91.0%	96.8%	63.9%	60.3%	51.3%	50.1%
	dyn.	93.5%	94.1%	62.8%	63.2%	63.1%	64.6%
20 m	kin.+	95.7%	97.6%	64.8%	40.2%	56.8%	50.0%
	dyn.	93.8%	95.4%	65.6%	52.0%	71.6%	63.5%

1. relative to the (non-enhanced) kin. model

drive models of the LandTamer. The enhanced differential drive model (diff. drive+) scales angular velocity linearly by the parameter α, which we calibrated via IPEM. The diff. drive+ model is significantly more accurate than the naïve diff. drive model, but less accurate than our enhanced 3D kinematic and dynamic models. Relative to the diff. drive+ model, position prediction errors are 32-56% smaller for the 3D kin.+ model, and 37-68% smaller for the 3D dyn. model (using converged parameter estimates).

Table 3.6: Pose prediction errors for differential drive models of the LandTamer (10 m horizon)

	asphalt		dirt		grass	
model	pos. err. (m)	yaw err. (rad)	pos. err. (m)	yaw err. (rad)	pos. err. (m)	yaw err. (rad)
diff. drive	3.9819	1.0411	4.8502	1.3279	4.6703	1.0515
diff. drive+ online	0.5208	0.0725	1.0925	0.1337	0.9565	0.1366
diff. drive+	0.3523	0.0257	0.8212	0.1044	0.8719	0.1096

RecBot Results. Prediction error results for the RecBot models on all three terrain types are presented in Tables 3.7 and 3.8. The RecBot results support the conclusions drawn from the LandTamer results: accounting for nonzero wheel slip significantly improves the accuracy of motion predictions, across all terrain types and horizons.

In our RecBot experiments, the enhanced kinematic model was marginally more accurate than the dynamic model. This is due in part to design choices. In the dynamic model, we chose all four wheels to have identical wheel-ground contact models; this keeps the model simple, but

Table 3.7: Pose prediction errors for the RecBot on three terrain types

		asphalt		dirt		grass	
horizon	model	pos. err. (m)	yaw err. (rad)	pos. err. (m)	yaw err. (rad)	pos. err. (m)	yaw err. (rad)
10 m	kin.	1.5985	0.1953	1.9500	0.3884	1.3843	0.1953
	kin.+ online	0.2723	0.0468	0.4945	0.0981	0.4119	0.0853
	kin.+	0.2096	0.0372	0.3742	0.0760	0.3364	0.0622
	dyn.	0.2818	0.0564	0.6325	0.1387	0.4947	0.0944
20 m	kin.	3.8325	0.3657	5.5051	0.7079	3.3847	0.3805
	kin.+	0.4762	0.0577	1.0067	0.1096	0.8972	0.1006
	dyn.	0.7686	0.0919	1.7544	0.1949	1.1547	0.1577

Table 3.8: Percent reductions in RecBot pose prediction errors

		asphalt		dirt		grass	
horizon	model	pos. err.	yaw err.	pos. err.	yaw err.	pos. err.	yaw err.
10 m	kin.+ online	83.0%	76.0%	74.6%	74.7%	70.2%	56.3%
	kin.+	86.9%	80.9%	80.8%	80.4%	75.7%	68.1%
	dyn.	82.4%	71.1%	67.6%	64.3%	64.3%	51.6%
	kin.+	87.6%	84.2%	81.7%	84.5%	73.5%	73.6%
	dyn.	79.9%	74.9%	68.1%	72.5%	65.9%	58.5%

limits the expression of asymmetry. Though the enhanced kinematic model's body-level slip parametrization in (2.48) and (2.49) has just six parameters, several terms express asymmetry. We attempted to make the dynamic model more expressive by allowing different contact model parameters for each wheel, but with so many parameters calibration was slow, and converged to local minima. Besides lower simulation cost, the enhanced kinematic model also has the advantage of requiring fewer parameters than a comparably expressive dynamic model.

In conclusion, accounting for nonzero wheel slip can dramatically improve predictive accuracy; this was consistently observed on multiple vehicles and terrain types. We do not claim that enhanced kinematics are more accurate than full dynamics in general, or even equally accurate. Clearly, our enhanced kinematic models cannot predict some extreme phenomena that dynamic models can (Section 2.5.3). However; our results suggest that under normal (near steady-state) operating conditions, our enhanced kinematic models *may* provide comparable accuracy to a dynamic model at significantly lower computational cost. Having fewer parameters may also make the enhanced kinematic model easier to calibrate.

3.4.3 3D Odometry Results

We have focused on motion planning applications for our WMR model formulations; however, they can also be useful for estimation. This section presents several 3D odometry experiments for the Zoë rover, of significantly longer duration than experiments in related work [75]. These experiments assess the usefulness of 3D kinematics vs. 2D kinematics, and of calibrating to data logs vs. manual calibration.

The Zoë rover was built with NASA support for the Life in the Atacama project [132]. Zoë has two articulated axles; each has a passive steering degree of freedom, and the rear axle has

a roll degree of freedom to keep the wheels in contact on uneven terrain [131]. Computation speeds for various Zoë rover models were benchmarked in Section 2.5.4. Photographs of Zoë are provided in Figure 3.15, and frame information is provided in Appendix E (Table E.2).

(a) (b)

Figure 3.15: (a) Photograph of Zoë in the Atacama desert in Chile. (b) Photograph of Zoë at the Robot City test site. Four ramp obstacles are also visible.

Inputs to the odometry include wheel velocities, axle angles, and attitude (roll/pitch angles); these were measured by encoders, potentiometers, and inclinometers respectively. Ground truth position was measured by a Leica robotic total station, which tracked a prism mounted on a pole at the center of the robot (Figure 3.15(b)). The prism had to be mounted high so that the camera mast would not break the line of sight between the prism and total station. Speeds were kept below 0.5 m/s to facilitate tracking.

First, we drove the Zoë rover in laps around a paved lot on the Carnegie Mellon University campus. During each lap the rover traversed two ramp obstacles. These ramps were 41 cm high by 179 cm long with a 61 cm flat middle section, as depicted in Figure E.2.

Zoë was commanded to drive straight over the ramps, but each perturbed it slightly off course. As a wheel climbs the ramp on one side, the rover articulates such that its body sways to the opposite side, affecting yaw and steer angles. The steering controller (having no 3D kinematic model) drives the steer angles back to zero, which causes a small turn. Figure 3.16 illustrates this process. These animation images are *not from a simulation*, but are a visualization of 3D odometry results. Notice that the estimated prism location (on top of the pole) tracks the measured path closely, even as it sways to the side.

We present results for three different models of the Zoë rover:

- *2D manual cal*: 2D kinematic model, manually calibrated
- *2D cal to data*: 2D kinematic model, calibrated to the data log using IPEM
- *3D cal to data*: 3D kinematic model (Section 2.2), calibrated to the data log using IPEM

Equations of the 2D kinematic model are presented in [112]; it ignores rear axle roll and assumes all contact angles are zero, restricting instantaneous motion to a plane. For the "manual cal" model, we set parameters via a manual procedure used on past expeditions to the Atacama. Wheel radius was measured with a meterstick, and coefficients in the linear equation relating

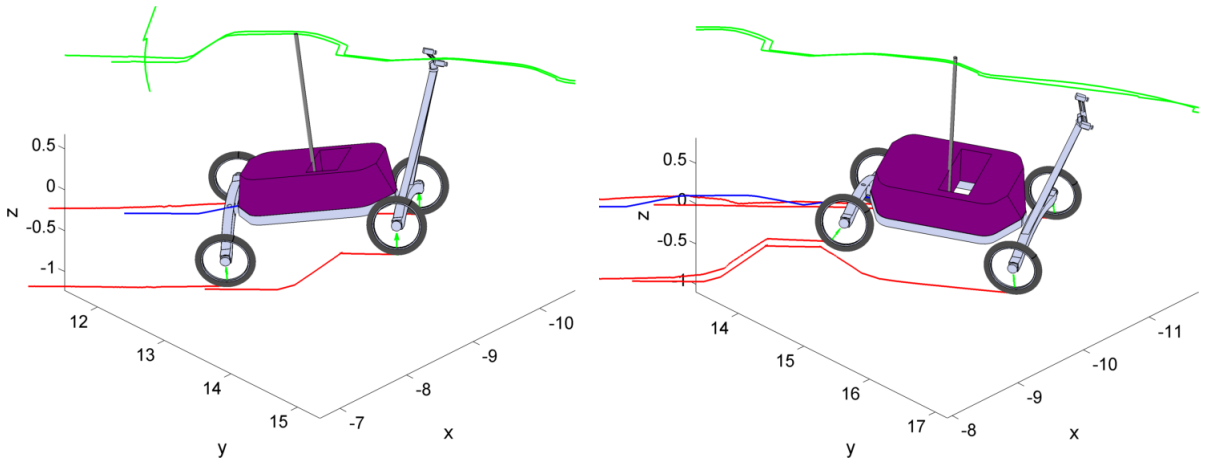


Figure 3.16: Animation of 3D odometry results as Zoë traverses a ramp. The path of the prism as measured by the total station is shown in green. The estimated paths of the body and wheel-ground contact frames are shown in blue and red, respectively.

potentiometer voltages to steer angles (3 total) were calculated from a few hand-collected data points; i.e. we used precision machined blocks to manually set the steer angles to 20° and -20°. For the "cal to data" models, these parameters were instead calibrated to the data logs via the same online IPEM algorithm used in Sections 3.4.1 and 3.4.2. Our presented results are for fully converged parameter estimates. No accounting for nonzero wheel slip due to gravitational or inertial force was required in these experiments, due to relatively level terrain and slow speeds.

We conducted two experiments on the paved lot at CMU, one in which only Zoë's left wheels traversed the ramps, and one in which only Zoë's right wheels traversed the ramps. Position and yaw errors throughout these experiments are shown in Figures 3.17 and 3.18.

Calibration to data logs via IPEM dramatically improved model accuracy relative to manual calibration. Using 3D kinematics also significantly improved accuracy relative to using 2D kinematics. When using 2D kinematics, spikes in yaw error occur when traversing ramps; these accumulate into a large positive error when the left wheels traverse ramps, and negative error when the right wheels traverse ramps. These errors are completely eliminated when using our 3D kinematics formulation.

The results are even more dramatic in a longer experiment conducted at the Robot City test site (Figures 3.15, 3.19). Here, Zoë's left wheels drove over four ramp obstacles per lap; the additional two ramps were 42 cm high by 134 cm long with no flat section. In 201 m of driving on uneven terrain, position error never exceeds 0.75 m and yaw error never exceeds 2.5°.

Interestingly, we noticed that 3D odometry assuming all contact angles (δ) are zero is no better than 2D odometry. Recall that contact angles specify the contact point location on the wheel surface, and $\delta = 0$ corresponds to the bottom of the wheel (Figure 2.3). In our experiment, we used prior knowledge of the obstacle geometry to help determine the contact angles. Iagnemma and Dubowsky [50] propose a method to estimate contact angles from wheel speeds and pitch rate, but it assumes no wheel slip. Some kinematic slip is unavoidable for our rover configuration on uneven terrain.

In general, contact angle estimation may require additional sensing such as an instrumented

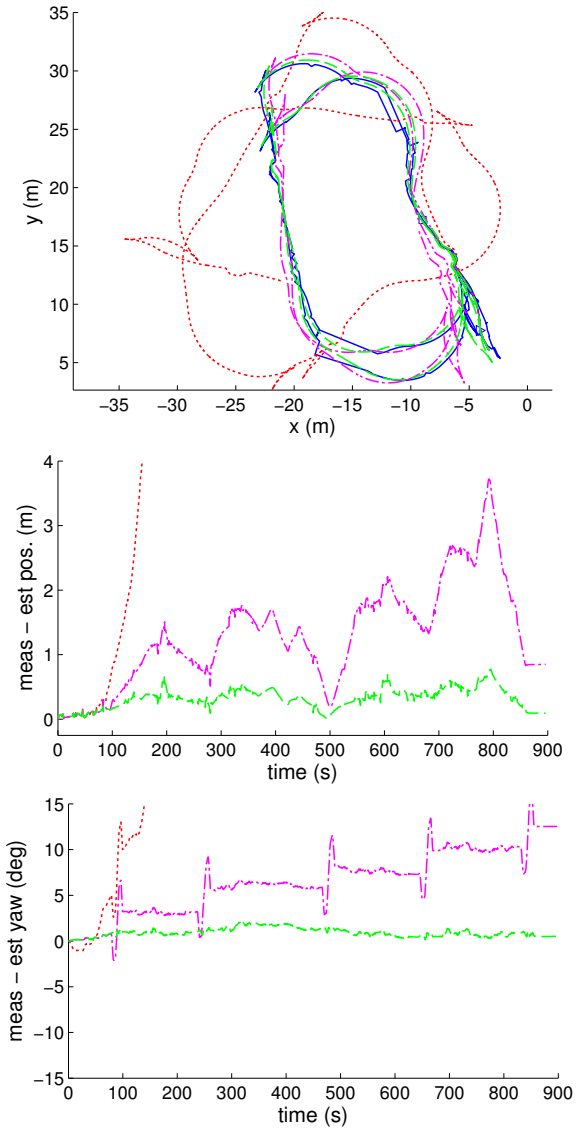


Figure 3.17: 3D odometry results as Zoë's left wheels traverse ramps (on paved lot at CMU). (Top) Measured and estimated paths of the rover for three models. (Middle) Estimated position error (Euclidean distance) vs. time. (Bottom) Estimated yaw error vs. time.

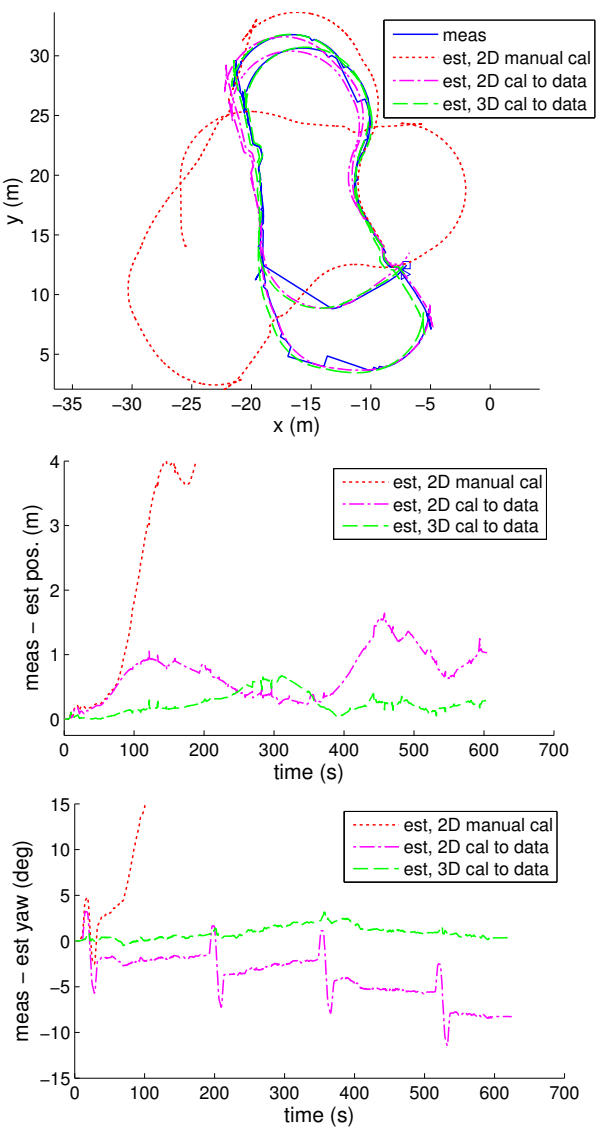


Figure 3.18: 3D odometry results as Zoë's right wheels traverse ramps

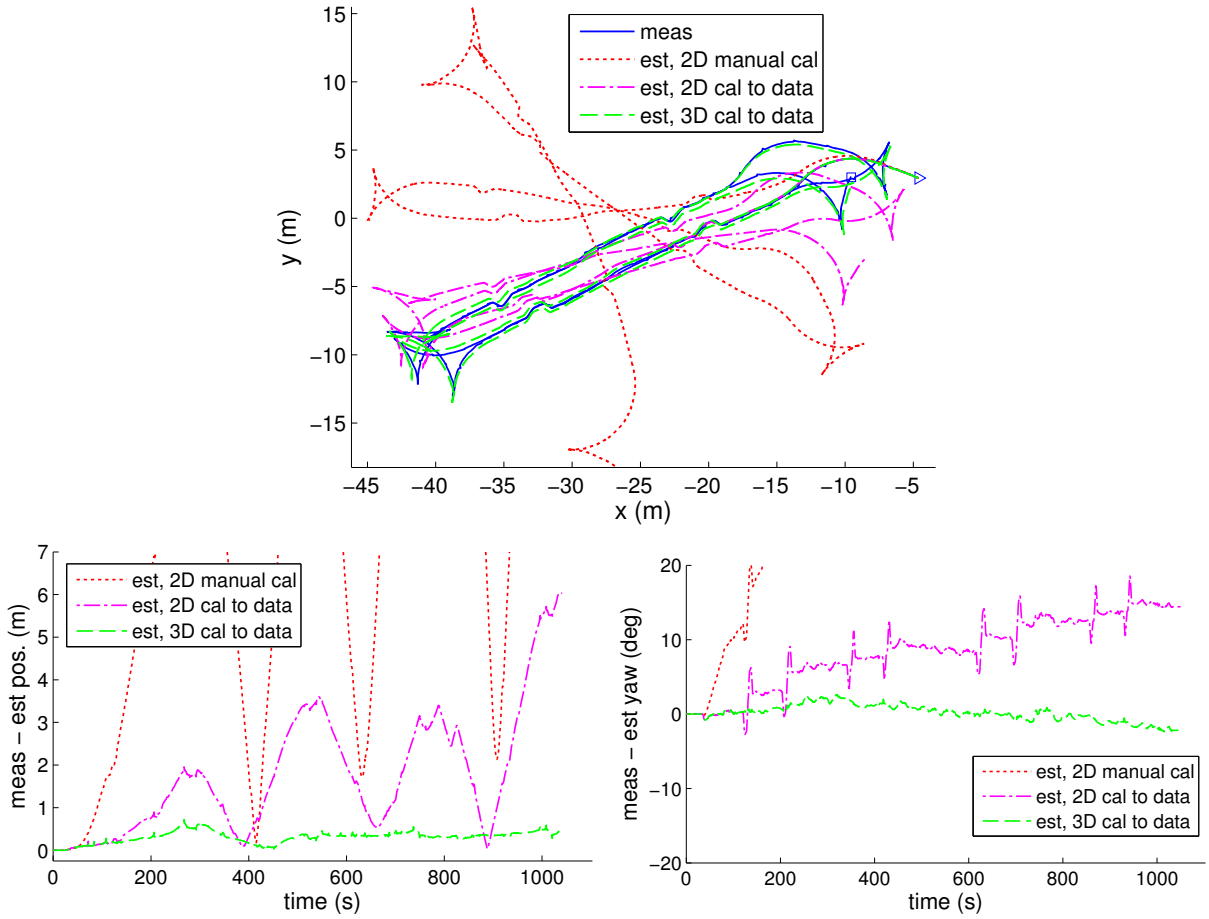


Figure 3.19: 3D odometry results for Zoë at Robot City

wheel. Lamon [74] describes a tactile wheel that senses contact angle via infrared proximity sensors mounted along the rim. Xu et al. [135] propose sensing contact angles visually. Given terrain geometry via perception, contact points my be solved for by collision detection. This is a necessity in simulation/planning contexts, but is also possible in estimation. Unfortunately perception was unavailable for our experiments on Zoë.

In conclusion, our 3D kinematic model formulation and IPEM calibration method both significantly improved the accuracy of odometry on the Zoë rover. Provided some means of sensing contact angles, 3D odometry should be used instead of 2D odometry for articulated WMRs on rough terrain.

Chapter 4

Conclusion

This chapter summarizes the contributions and results in this thesis (Section 4.1) and identifies areas for future work (Section 4.2).

4.1 Contributions and Results

Much effort has been spent on developing model-based algorithms for WMR motion planning, control, and estimation. The effectiveness of these algorithms depends critically on the fidelity and speed of their underlying motion models, but considerably less effort has been spent (in recent years) on general WMR modeling methods. Producing dynamic models for today's highly mobile platforms remains a challenge. This thesis advances state of the art in both the *formulation* and *calibration* of WMR models.

Formulation. Per the requirements in Section 1.1.1, this thesis presents new formulations of WMR kinematics and dynamics that are high-fidelity, general, modular, and fast.

Some have recently published general methods for the derivation of 3D WMR kinematics [125][126][127][25]; however, we are the first to combine general kinematics derivation with the use of DAEs and Baumgarte's stabilization method for constrained, drift-free motion prediction (Section 2.2). We also enhanced the kinematics in a novel way to predict nonzero wheel slip; our parametrization of slip over gravitational, inertial, and dissipative forces matches experimentally observed behaviors (Section 2.3).

Some have developed 3D dynamics models for *particular* platforms, most notably ROAMS for the Mars Exploration Rovers [55][118][48], but we provide the first general method to model the 3D dynamics of *any* WMR (Section 2.4). Our method leverages efficient recursive algorithms, originally designed to model the dynamics of manipulators. Typically WMR dynamics are modeled using unconstrained ordinary differential equations (ODEs), but we developed a unique DAE formulation. In contrast to Open Dynamics Engine, which uses constraints to approximate Coulomb friction at contact points, we can enforce *nonlinear* models of wheel-terrain interaction via force-balance optimization (Section 2.4.3).

Simulation tests show our "stabilized DAE" method of kinematic motion prediction to be more accurate and much faster than the common unconstrained method, due to the elimination of terrain fitting (Section 2.5.1). Tests also show our DAE-based method for dynamic modeling

to be less stiff than the common ODE-based method. Our constrained method allows large, fixed integration steps to be taken without compromising stability, which greatly speeds up computation (Section 2.5.2). Our dynamic models can predict loss of traction phenomena that Open Dynamics Engine models cannot, as well as liftoff/rollover (Section 2.5.3).

Benchmark tests demonstrated that our kinematic and dynamic models can be simulated 1K-10K$\times$ faster than real-time on an ordinary PC (Section 2.5.4). Our dynamics formulation is faster than Open Dynamics Engine when using the same wheel-ground contact model, and comparably fast when using nonlinear empirical and terramechanics-based models. Our dynamic simulations run orders of magnitude faster than those of comparable fidelity in related work on MPP [53].

Calibration. Per Section 1.1.2, this thesis also presents a novel Integrated Prediction Error Minimization (IPEM) method of calibrating model parameters that is general, convenient, online, and evaluative.

System dynamics are typically identified by minimizing the error of differential equation (DE) residuals, which compare instantaneous measured vs. predicted output. In IPEM, we instead minimize residuals formed by integrating the system dynamics over an interval. Benefits of this approach include reduced sensing requirements, better observability, and accurate prediction over a longer horizon (Section 3.1.1).

Our work on IPEM is novel in several ways. First, most related work is limited to 2D odometry calibration for a particular platform [83][84][5][6][22]. In contrast, our IPEM method applies to *any* WMR model, including our own high-fidelity, kinematic and dynamic models. In addition, this thesis uniquely analyzes the benefits of calibrating to integrated prediction errors rather than instantaneous output errors. This thesis also provides a new approach to uncertainty evaluation, using unified models of systematic and stochastic error propagation (Section 3.2). Finally, this thesis provides unprecedented experimental validation on multiple vehicles and terrain types.

Our experiments on Crusher show that parameter estimates converge quickly during online calibration. Furthermore, our calibrated stochastic model accurately characterizes the uncertainty of motion predictions (Section 3.4.1). Experiments on the LandTamer and RecBot show dramatic improvement in predictive accuracy when accounting for nonzero wheel slip. They also show that, under normal conditions, our enhanced kinematic model can provide comparable accuracy to a dynamic model at a fraction of the computational cost (Section 3.4.2). Finally, experiments on the Zoë rover demonstrate a major improvement in odometry accuracy when calibrating to data logs via IPEM and accounting for 3D articulations (Section 3.4.3).

In conclusion, we have provided powerful new tools for modeling WMRs and thoroughly evaluated them in both simulations and physical experiments. Using these tools, one can construct a realistic, 3D kinematic/dynamic model for any articulated WMR with just a few lines of code. One can conveniently calibrate the model parameters online during normal operation, optimizing predictive accuracy over extended horizons. Finally, one can forward simulate the model *3-4 orders of magnitude* faster than real-time. This research has many applications, but the most immediate are in model predictive planning; the effectiveness of MPP algorithms like those in [47][95][53] increases with the speed and accuracy of their underlying motion models.

4.2 Future Work

There are several possibilities for extending this research:

This thesis addresses the tradeoff between model fidelity and computation speed observed in related work. While our new formulations ameliorate this tradeoff, it cannot be completely eliminated. Predicting some maneuvers requires a full dynamic model, which is necessarily more expensive than a kinematic model (Section 2.5.3); however, sometimes a cheaper enhanced kinematic model is sufficiently accurate (Section 3.4.2). One possibility for future work is to automate the selection of the appropriate model fidelity (i.e. dynamic, dynamic with approximation, enhanced kinematic, 3D kinematic, 2D kinematic). In MPP, one might even dynamically switch between models to reduce computation while still guaranteeing feasibility.

Computational efficiency might be further improved by fully exploiting opportunities for vectorization and parallelization. Simulation is embarrassingly parallel in the number of trajectories, but parallelization might even be possible when simulating a single trajectory. Calibration could be more efficient if using a derivative-free optimization algorithm, or if computing Jacobians some other way than finite differences.

For greater functionality, our models could be extended to handle other modes of locomotion besides wheels, such as tracks or legs. In addition, they could be extended to handle contact forces experienced during mobile manipulation. Finally, our IPEM calibration method could be extended for lifelong learning, or for the sharing of learned terrain properties between vehicles.

Our model formulation and calibration methods have immediate applications in planning, control, and estimation. We have released open source MATLAB and C++ libraries implementing these methods, along with documentation, examples, and test data sets [109]; this should facilitate their use and provide opportunities for collaboration. We plan to maintain and hopefully improve these code libraries in the future. Beyond immediate applications, we hope these methods will make possible new research directions towards fully exploiting WMR mobility.

Bibliography

[1] Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Y Ng, and Sebastian Thrun. Discriminative training of Kalman filters. In *Robotics: Science and Systems*, 2005. 3.1.2, 3.2.4, 3.2.4

[2] J. Agullo, S. Cardona, and J. Vivancos. Kinematics of vehicles with directional sliding wheels. *Mechanical Machine Theory*, 22(4):295–301, 1987. 2.1.1, 2.1

[3] Bernt M. Åkesson, John Bagterp Jørgensen, Niels Kjølstad Poulsen, and Sten Bay Jørgensen. A tool for kalman filter tuning. In *Computer Aided Process Engineering*, 2007. 3.1.2

[4] J. C. Alexander and J. H. Maddocks. On the kinematics of wheeled mobile robots. *International Journal of Robotics Research*, 8(5):15–27, 1989. 2.1.1, 2.1

[5] Gianluca Antonelli, Stefano Chiaverini, and Giuseppe Fusco. A calibration method for odometry of mobile robots based on the least-squares technique: Theory and experimental validation. *Transactions on Robotics*, 21(5):994–1004, 2005. 3.1.3, 3.1, 3.1.4, 4.1

[6] Gianluca Antonelli, Fabrizio Caccavale, Flavio Grossi, and Alessandro Marino. A non-iterative and effective procedure for simultaneous odometry and camera calibration for a differential drive mobile robot based on the singular value decomposition. *Intelligent Service Robotics*, 3:163–173, June 2010. 3.1.3, 3.1, 4.1

[7] R. Balakrishna and Ashitava Ghosal. Modeling of slip for wheeled mobile robots. *Transactions on Robotics and Automation*, 11(1), 1995. 2.1.2, 2.1, 2.4.1, 2.5.2

[8] J. Bob Balaram. Kinematic observers for articulated rovers. In *International Conference on Robotics and Automation*. IEEE, 2000. 2.1.1, 2.1.1, 2.1

[9] David Baraff. Linear-time dynamics using Lagrange multipliers. In *ACM SIGGRAPH*, 1996. 2.2.1

[10] Joachim Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering*, 1:1–16, 1972. 2.2.5

[11] M. G. Bekker. *Off-the-road locomotion*. University of Michigan Press, Ann Arbor, MI, 1960. 2.4.3, C

[12] Faïz Benamar, Philippe Bidaud, and Frédéric Le Menn. Generic differential kinematic modeling of articulated mobile robots. *Mechanism and Machine Theory*, 45(7):997–1012, July 2010. 2.1.1, 2.1

[13] Jur van den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty

using differential dynamic programming in belief space. In *International Symposium on Robotics Research*, 2011. 3.1.3

[14] M. C. Best, A P Newton, and S Tuplin. The identifying extended Kalman filter: parametric system identification of a vehicle handling model. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, 221(1):87–98, January 2007. 2.1.2

[15] Geoge W. Bohrnstedt and Arthur S. Goldberger. On the exact covariance of products of random variables. *Journal of the American Statistical Association*, 64(328):1439–1442, 1969. 3.2.2

[16] J. Borenstein. Control and kinematic design of multi-degree-of freedom mobile robots with compliant linkage. *Transactions on Robotics and Automation*, 11(1):21–35, 1995. 2.1.1, 2.1

[17] Johann Borenstein and Liqiang Feng. Measurement and correction of systematic odometry errors in mobile robots. *Transactions on Robotics and Automation*, 12(6):869–880, 1996. 3.1.3, 3.1

[18] Raymond Brach and Matthew Brach. The Tire-Force Ellipse (Friction Ellipse) and Tire Characteristics. *SAE Technical Paper*, (2011-01-0094), 2011. 2.5.3, 2.15, C

[19] Raymond M. Brach and R. Matthew Brach. Tire models for vehicle dynamic simulation and accident reconstruction. *SAE Technical Paper*, (2009-01-0102), 2009. 2.3, 2.4.3, 2.5.2, 2.5.4, C, C

[20] Thomas Burke and Hugh F. Durrant-Whyte. Kinematics for modular wheeled mobile robots. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 1993. 2.1.1, 2.1

[21] Guy Campion, Georges Bastin, and Brigitte D'Andréa-Novel. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *Transactions on Robotics*, 12(1):47–62, 1996. 2.1.1

[22] Andrea Censi, Luca Marchionni, and Giuseppe Oriolo. Simultaneous maximum-likelihood calibration of odometry and sensor parameters. In *International Conference on Robotics and Automation*. IEEE, 2008. 3.1.3, 3.1, 4.1

[23] Nilanjan Chakraborty and Ashitava Ghosal. Kinematics of wheeled mobile robots on uneven terrain. *Mechanism and Machine Theory*, 39(12):1273–1287, December 2004. 2.1.1, 2.1, 2.4.3, 2.5.1

[24] Yong Chang, Dalong Tan, Hongguang Wang, and Shugen Ma. Kinematics analysis of a six-wheeled mobile robot. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, October 2006. 2.1.1, 2.1

[25] Yong Chang, Shugen Ma, Hongguang Wang, and Dalong Tan. A kinematic modeling method for a wheeled mobile robot. In *International Conference on Mechatronics and Automation*, pages 1100–1105. IEEE, August 2009. 2.1.1, 2.1, 2.1.3, 2.2.1, 4.1

[26] B J Choi and S V Sreenivasan. Gross motion characteristics of articulated mobile robots with pure rolling capability on smooth uneven surfaces. *Transactions on Robotics*, 15(2):

340–343, 1999. 2.1.1, 2.1, 2.4.3, 2.5.1

[27] Kok Seng Chong and Lindsay Kleeman. Accurate odometry and error modelling for a mobile robot. In *International Conference on Robotics and Automation*. IEEE, 1997. 3.1.3, 3.1

[28] B. D'Andréa-Novel, G. Bastin, and G. Campion. Modelling and control of non-holonomic wheeled mobile robots. In *International Conference on Robotics and Automation*. IEEE, 1991. 2.1.2, 2.1

[29] Alessandro De Luca and Giuseppe Oriolo. Chapter 7: Modeling and control of nonholonomic mechanical systems. In *Kinematics and Dynamics of Multi-Body Systems*, pages 277–342. Springer Verlag, 1995. 2.1.2

[30] Liang Ding, Haibo Gao, Zongquan Deng, Peilin Song, and Rongqiang Liu. Design of comprehensive high-fidelity/high-speed virtual simulation system for lunar rover. In *Conference on Robotics, Automation and Mechatronics*. IEEE, September 2008. 2.1.2, 2.1

[31] Liang Ding, Kazuya Yoshida, Keiji Nagatani, Haibo Gao, and Zongquan Deng. Parameter identification for planetary soil based on a decoupled analytical wheel-soil interaction terramechanics model. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2009. 2.4.3, 3.1.3

[32] Evan Drumwright, John Hsu, Nathan Koenig, and Dylan Shell. Extending Open Dynamics Engine for robotics simulation. In *Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2010. 2.1.2, 2.5.4

[33] Hoai Nghia Duong and Ioan Dore Landau. On test horizon for model validation by output error. *Transactions on Automatic Control*, 39(1):102–106, 1994. 3.2.1

[34] Vijay Eathakota, Gattupalli Aditya, and Madhava Krishna. Quasi-static motion planning on uneven terrain for a wheeled mobile robot. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2011. 2.1.1, 2.1.2, 2.1, 2.5.1

[35] Marcello Farina and Luigi Piroddi. Some convergence properties of multi-step prediction error identification criteria. In *Conference on Decision and Control*. IEEE, 2008. 3.1.2

[36] Roy Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, Boston/Dordrecht/Lancaster, 1987. 1.3, 2.4.1, B

[37] Roy Featherstone and David Orin. Robot dynamics: Equations and algorithms. In *International Conference on Robotics and Automation*. IEEE, 2000. 1.3, 2.4.1

[38] Qiushi Fu and Venkat Krovi. Articulated wheeled robots: Exploiting reconfigurability and redundancy. In *ASME Dynamic Systems and Control Conference*, pages 653–660, 2008. 2.1.1, 2.1

[39] Aditya Gattupalli, Vijay Eathakota, Arun Kumar Singh, and Madhava Krishna. A simulation framework for evolution on uneven terrains for synchronous drive robot. *Advanced Robotics*, 27(8):641–654, 2013. 2.1.1, 2.1, 2.5.1

[40] Puneet Goel, Stergios I. Roumeliotis, and Gaurav S. Sukhatme. Robust localization using relative and absolute position estimates. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 1999. 3.1.3, 3.1

[41] Juan P. Gonzalez and Anthony Stentz. Replanning with uncertainty in position: Sensor updates vs. prior map updates. In *International Conference on Robotics and Automation*. IEEE, 2008. 3.1.3

[42] Sven Gowal, Yizhen Zhang, and Alcherio Martinoli. A realistic simulator for the design and evaluation of intelligent vehicles. In *IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2010. 2.1.2, 2.4.3, 2.5.4

[43] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Geometric numerical integration illustrated by the Störmer-Verlet method. *Acta Numerica*, 12:399–450, 2003. 2.4.4

[44] P. Heiskanen, S. Heikkilä, and A. Halme. Development of a dynamic mobile robot simulator for astronaut assistance. In *10th ESA Workshop for Advanced Space Technologies for Robotics and Automation*, 2008. 2.1.2, 2.5.4

[45] Daniel M. Helmick, Stergios I. Roumeliotis, Yang Cheng, Daniel S. Clouse, Max Bajracharya, and Larry H. Matthies. Slip-compensated path following for planetary exploration rovers. *Advanced Robotics*, 20(11):1257–1280, November 2006. 2.1.1

[46] Thomas M. Howard and Alonzo Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, 26(2):141–166, February 2007. 2.1.1, 2.1.2, 2.2.4, 2.3, 2.5.1, 2.5.1

[47] Tom Howard. Adaptive model-predictive motion planning for navigation in complex environments. Technical Report CMU-RI-TR-09-32, Carnegie Mellon University, Robotics Institute, 2009. PhD thesis. 1.1.1, 2.1.2, 4.1

[48] Terry Huntsberger, Abhi Jain, Jonathan Cameron, Gail Woodward, David Myers, and Garrett Sohl. Characterization of the ROAMS simulation environment for testing rover mobility on sloped terrain. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2008. 2.1.2, 2.1, 4.1

[49] Karl Iagnemma. *Rough-terrain mobile robot planning and control with application to planetary exploration*. PhD thesis, Massachusetts Institute of Technology, 2001. 3.1.3

[50] Karl Iagnemma and Steven Dubowsky. Vehicle wheel-ground contact angle estimation: with application to mobile robot traction control. In *International Symposium on Advances in Robot Kinematics*, 2000. 3.4.3

[51] Karl Iagnemma, Hassan Shibly, Adam Rzepniewski, and Steven Dubowsky. Planning and control algorithms for enhanced rough-terrain rover mobility. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2001. 2.1.2, 2.2.4

[52] Genya Ishigami, Akiko Miwa, Keiji Nagatani, and Kazuya Yoshida. Terramechanics-based model for steering maneuver of planetary exploration rovers on loose soil. *Journal of Field Robotics*, 24(3):233–250, 2007. 2.1.2, 2.4.3, 2.5.4, 2.5.4, 3.1.3, C, C.2, C.4, C, C, C.5

[53] Genya Ishigami, Keiji Nagatani, and Kazuya Yoshida. Path planning and evaluation for planetary rovers based on dynamic mobility index. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, September 2011. 2.1.2, 2.5.4, 4.1

[54] ISO 8855:2011. Road vehicles – Vehicle dynamics and road-holding ability – Vocabulary,

2011. 2.2.1, A

[55] A. Jain, J. Guineau, C. Lim, W. Lincoln, M. Pomerantz, G. Sohl, and R. Steele. ROAMS: Planetary surface rover simulation environment. *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2003. 2.1.2, 2.1, 4.1

[56] Zhenzhong Jia, William Smith, and Huei Peng. Fast computation of wheel-soil interactions for safe and efficient operation of mobile robots. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2011. 2.4.3, C

[57] John Bagterp Jørgensen and Sten Bay Jørgensen. MPC-relevant prediction-error identification. In *American Control Conference*, 2007. 3.1.2

[58] John Bagterp Jørgensen and Sten Bay Jørgensen. Comparison of prediction-error-modelling criteria. In *American Control Conference*, 2007. 3.1.2

[59] Yutaka Kanayama. Two dimensional wheeled vehicle kinematics. In *International Conference on Robotics and Automation*. IEEE, 1994. 2.1.1, 2.1

[60] Alonzo Kelly. Linearized error propagation in odometry. *International Journal of Robotics Research*, 23(2):179–218, February 2004. 3.1.3, 3.1.4

[61] Alonzo Kelly. Fast and easy systematic and stochastic odometry calibration. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2004. 3.1.3, 3.1.3, 3.1

[62] Alonzo Kelly. A vector algebra formulation of kinematics of wheeled mobile robots. Technical Report CMU-RI-TR-10-33, Carnegie Mellon University, Robotics Institute, 2010. 2.1.1, 2.1, 2.1.3

[63] Alonzo Kelly. *Mobile Robots: Mathematics, Models, and Methods*. Cambridge University Press, 2013. 4.1, A, A

[64] Alonzo Kelly and Neal Seegmiller. A vector algebra formulation of mobile robot velocity kinematics. In *Field and Service Robotics*, 2012. 2.1.1, 2.1, 2.1.3, 2.2.2, 2.3

[65] Alonzo Kelly and Neal Seegmiller. Recursive kinematic propagation for wheeled mobile robots. *International Journal of Robotics Research*, 2014. to appear. 2.1.1, 2.2.2

[66] Alonzo Kelly and Neal Seegmiller. Modeling of wheeled mobile robots as differential-algebraic systems. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2014. 2.1.3

[67] Dong Sung Kim, Hyun Chul Lee, and Wook Hyun Kwon. Geometric kinematics modeling of omni-directional autonomous mobile robot and its applications. In *International Conference on Robotics and Automation*. IEEE, 2000. 2.1.1

[68] Dong-sung Kim, Wook Hyun Kwon, and Hong Sung Park. Geometric kinematics and applications of a mobile robot. *International Journal of Control, Automation, and Systems*, 1(3):376–384, 2003. 2.1.1

[69] Wheekuk Kim, Byung-Ju Yi, and Dong Jin Lim. Kinematic modeling of mobile robots by transfer method of augmented generalized coordinates. *Journal of Robotic Systems*, 21 (6):301–322, June 2004. 2.1.1, 2.1

[70] Young Yong Kim, Mun-Ho Jeong, and Dong Joong Kang. Mobile robot calibration. In *International Conference on Control, Automation, and Systems*. IEEE, 2013. 3.1.3, 3.1

[71] Ross Knepper. On the fundamental relationships among path planning alternatives. Technical Report CMU-RI-TR-11-19, Carnegie Mellon University, Robotics Institute, 2011. PhD thesis. 2.5.4, 4.1

[72] Oleksiy V. Korniyenko, Mohammed S. Sharawi, and Daniel N. Aloi. Neural network based approach for tuning Kalman filter. In *International Conference on Electro Information Technology*. IEEE, 2005. 3.1.2

[73] Rainer Kümmerle, Giorgio Grisetti, and Wolfram Burgard. Simultaneous parameter calibration, localization, and mapping. *Advanced Robotics*, 26(17):2021–2041, December 2012. 3.1.3, 3.1

[74] Pierre Lamon. *3D-Position Tracking and Control for All-Terrain Robots*, volume 43 of *Tracts in Advanced Robotics*. Springer, 2008. 2.1.2, 3.4.3

[75] Pierre Lamon and Roland Siegwart. 3D position tracking in challenging terrain. *International Journal of Robotics Research*, 26(2):167–186, February 2007. 2.1.1, 2.1.2, 2.1, 2.5.4, 3.4.3

[76] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001. 2.1.2

[77] Frederic Le Menn, Philippe Bidaud, and Faiz Ben Amar. Generic differential kinematic modeling of articulated multi-monocycle mobile robots. In *International Conference on Robotics and Automation*. IEEE, 2006. 2.1.1, 2.1

[78] Randel A. Lindemann and Chris J. Voorhees. Mars Exploration Rover mobility assembly design, test and performance. In *International Conference on Systems, Man and Cybernetics*. IEEE, 2005. 1

[79] Lennart Ljung. *System Identification - Theory for the User*. Prentice Hall, 1987. 3.1.2

[80] E. Lucet, C. Grand, D. Sallé, and P. Bidaud. Dynamic sliding mode control of a four-wheel skid-steering vehicle in presence of sliding. In *Symposium on Theory and Practice of Robots and Manipulators (RoManSy)*, 2008. 2.1.1

[81] John Y. S. Luh, Michael W. Walker, and Richard P. C. Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 02(1980):69–76, 1980. 2.1.1

[82] Anthony Mandow, Jorge L. Martinez, Jesus Morales, Jose L. Blanco, Alfonso Garcia-Cerezo, and Javier Gonzalez. Experimental kinematics for wheeled skid-steer mobile robots. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2007. 2.4.3, 3.4.1

[83] Agostino Martinelli. The odometry error of a mobile robot with a synchronous drive system. *Transactions on Robotics*, 18(3):399–405, 2002. 3.1.3, 3.1.3, 3.1, 3.1.4, 4.1

[84] Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. Simultaneous localization and odometry self calibration for mobile robot. *Autonomous Robots*, 22(1):75–85, September 2007. 3.1.3, 3.1, 4.1

[85] Matthew T. Mason. *Mechanics of Robotic Manipulation*. The MIT Press, Cambridge, MA, 2001. A

[86] Scott McMillan and David E Orin. Forward dynamics of multilegged vehicles using the composite rigid body method. In *International Conference on Robotics and Automation*. IEEE, 1998. 2.1.2, 2.4.1

[87] Mechanical Simulation Corporation. CarSim. `https://www.carsim.com/products/carsim/`, 2014. Accessed: 2014-08-07. 2.1.2

[88] Patrick Muir. Modeling and control of wheeled mobile robots. Technical Report CMU-RI-TR-88-20, Carnegie Mellon University, Robotics Institute Tech. Report, August 1988. PhD thesis. 2.1.2, 2.1

[89] Patrick Muir and Charles Neuman. Kinematic modeling of wheeled mobile robots. Technical Report CMU-RI-TR-86-12, Carnegie Mellon University, Robotics Institute, June 1986. 2.1.1, 2.1, 2.3

[90] S. Nandy, S. N. Shome, R. Somani, T. Tanmay, G. Chakraborty, and C. S. Kumar. Detailed slip dynamics for nonholonomic mobile robotic system. In *International Conference on Mechatronics and Automation*. IEEE, August 2011. 2.1.2, 2.1

[91] G. De Nicolao. System identification: Problems and perspectives. In *12th Workshop on Qualitative Reasoning*, June 1997. 3.1.2, 3.1.2

[92] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, 1999. D

[93] Kris Osborn and Andrew Kerbrat. Army testing rugged, autonomous robot vehicle. `http://www.army.mil/article/40223/army-testing-rugged-autonomous-robot-vehicle/`, June 2010. 1

[94] Hans B. Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle System Dynamics*, 21(Supplement 001):1–18, 1992. 2.4.3

[95] Mihail Pivtoraiko. Differentially constrained motion planning with state lattice motion primitives. Technical Report CMU-RI-TR-12-07, Carnegie Mellon University, Robotics Institute, 2012. PhD thesis. 4.1

[96] Mihail Pivtoraiko and Alonzo Kelly. Kinodynamic motion planning with state lattice motion primitives. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2011. 2.1.2

[97] Thomas D Powell. Automated tuning of an extended Kalman filter using the downhill simplex algorithm. *Journal of Guidance, Control, and Dynamics*, 25(5):901–908, 2002. 3.1.2

[98] Roya Rahbari, Barrie W. Leach, and Clarence W. de Silva. Adaptive tuning of a Kalman filter using the fuzzy integral for an intelligent navigation system. In *International Symposium on Intelligent Control*. IEEE, 2002. 3.1.2

[99] R. Rajagopalan. A generic kinematic formulation for wheeled mobile robots. *Journal of Robotic Systems*, 14(2):77–91, February 1997. 2.1.1, 2.1

[100] G. Rodriguez, K. Kreutz-Delgado, and A. Jain. A spatial operator algebra for manipulator modeling and control. *International Journal of Robotics Research*, 10:371–381, 1991. 2.1.2

[101] Mitchell M. Rohde, Justin Crawford, Matthew Toschlog, Karl D. Iagnemma, Gaurav Kewlani, Christopher L. Cummins, Randolph A. Jones, and David A. Horner. An interactive , physics-based unmanned ground vehicle simulator leveraging open source gaming technology: Progress in the development and application of the Virtual Autonomous Navigation Environment (VANE) desktop. In *SPIE*, 2009. 2.1.2, 2.5.4

[102] N. Roy and S. Thrun. Online self-calibration for mobile robots. In *International Conference on Robotics and Automation*. IEEE, 1999. 3.1.3, 3.1

[103] Alexander Rudolph. Quantification and estimation of differential odometry errors in mobile robotics with redundant sensor information. *International Journal of Robotics Research*, 22(2):117–128, February 2003. 3.1.3, 3.1

[104] Manika Saha, Bhaswati Goswami, and Ratna Ghosh. Two novel metrics for determining the tuning parameters of the Kalman Filter. *arXiv:1110.3895*, 2011. 3.1.2

[105] Subir Kumar Saha and Jorge Angeles. Kinematics and dynamics of a three-wheeled 2-DOF AGV. In *International Conference on Robotics and Automation*. IEEE, 1989. 2.1.1, 2.1.2, 2.1, 2.4.2

[106] N. Seegmiller, F. Rogers-Marcovitz, G. Miller, and A. Kelly. A unified perturbative dynamics approach to vehicle model identification. In *International Symposium on Robotics Research*, August 2011. 3.1.4

[107] N. Seegmiller, F. Rogers-Marcovitz, and A. Kelly. Online calibration of vehicle powertrain and pose estimation parameters using integrated dynamics. In *International Conference on Robotics and Automation*. IEEE, 2012. 3.1.4, 3.4

[108] N. Seegmiller, F. Rogers-Marcovitz, G. Miller, and A. Kelly. Vehicle model identification by integrated prediction error minimization. *International Journal of Robotics Research*, 32(8):912–931, July 2013. 3.1, 3.1.4, 3.2.1, 3.2.2, 3.4.1

[109] Neal Seegmiller. WMR Dynamics Engine software library. `https://github.com/nealseegmiller/wmrde.git`, 2014. 2.5, 4.2

[110] Neal Seegmiller and Alonzo Kelly. Modular dynamic simulation of wheeled mobile robots. In *Field and Service Robotics*, 2013. 2.1, 2.1.3

[111] Neal Seegmiller and Alonzo Kelly. Enhanced 3D kinematic modeling of wheeled mobile robots. In *Robotics: Science and Systems*, 2014. 2.1, 2.1.3, 3.1, 3.1.4

[112] Neal Seegmiller and David Wettergreen. Control of a passively steered rover using 3-D kinematics. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2011. 2.1, 3.4.3

[113] Dong Hun Shin and Kyung Hoon Park. Velocity kinematic modeling for wheeled mobile robots. In *International Conference on Robotics and Automation*. IEEE, 2001. 2.1.1, 2.1

[114] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*, chapter 14. Springer, Berlin, Heidelberg, 2008. 3.1.1

[115] Naim Sidek and Nilanjan Sarkar. Dynamic modeling and control of nonholonomic mobile robot with lateral slip. In *Third International Conference on Systems*. IEEE, April 2008. 2.1.2, 2.1

[116] Russell Smith. Open Dynamics Engine v0.5 User Guide. `http://ode.org/ode-latest-userguide.html`, 2006. Accessed: 2014-08-09. 2.4.3, 2.5.4, 4.1

[117] Torsten Söderström and Petre Stoica. *System Identification*. Prentice Hall, 1989. 3.1.2

[118] Garett Sohl and Abhinandan Jain. Wheel-terrain contact modeling in the ROAMS planetary rover simulation. In *ASME International Design Engineering Technical Conferences*, 2005. 2.1.2, 2.1, 2.4.1, 2.5.2, 3.1.3, 4.1

[119] Selim Solmaz, Mehmet Akar, Robert Shorten, and Jens Kalkkuhl. Realtime multiple-model estimation of center of gravity position in automotive vehicles. *Vehicle System Dynamics*, 46, 2008. 2.1.2

[120] Jae-Bok Song and Kyung-Seok Byun. Design and control of a four-wheeled omnidirectional mobile robot with steerable omnidirectional wheels. *Journal of Robotic Systems*, 21(4):193–208, April 2004. 2.1.2, 2.1

[121] Xiaojing Song, Zibin Song, Lakmal D. Seneviratne, and Kaspar Althoefer. Optical flow-based slip and velocity estimation technique for unmanned skid-steered vehicles. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2008. 2.1.1

[122] Robert F. Stengel. *Optimal Control and Estimation*. Dover, New York, 1994. 3.2.1, 3.2.1, 3.2.1, 3.2.1

[123] D. Stewart and J.C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with Coulomb friction. In *International Conference on Robotics and Automation*. IEEE, 1996. 2.1.2, 2.4.2

[124] S. Sugano, Q. Huang, and I. Kato. Stability criteria in controlling mobile robotic systems. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, July 1993. 2.3

[125] M. Tarokh and G.J. McDermott. Kinematics modeling and analyses of articulated rovers. *Transactions on Robotics*, 21(4):539–553, August 2005. 2.1, 2.1.1, 2.1, 2.2.4, 2.2.4, 2.3, 2.5.1, 2.5.1, 4.1

[126] Mahmoud Tarokh and Gregory Mcdermott. A systematic approach to kinematics modeling of high mobility wheeled rovers. In *International Conference on Robotics and Automation*. IEEE, 2007. 2.1.1, 2.1, 4.1

[127] Mahmoud Tarokh, Huy Dang Ho, and Antonios Bouloubasis. Systematic kinematics analysis and balance control of high mobility rovers over rough terrain. *Robotics and Autonomous Systems*, 61(1):13–24, January 2013. 2.1.1, 2.1, 2.1.3, 2.2.1, 4.1

[128] Yu Tian, Naim Sidek, and Nilanjan Sarkar. Modeling and control of a nonholonomic wheeled mobile robot with wheel slip dynamics. In *Symposium on Computational Intelligence in Control and Automation*. IEEE, 2009. 2.1.2, 2.1

[129] Yutaka Tomita, Ad A. H. Damen, and Paul M. J. Van den Hof. Equation error versus output error methods. *Ergonomics*, 35(5):551–564, 1992. 3.1.2

[130] Chris Urmson et al. Autonomous driving in urban environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. 1.1

[131] Michael Wagner, Stuart Heys, David Wettergreen, James Teza, Dimitrios Apostolopoulos, and George Kantor. Design and control of a passively steered, dual axle vehicle. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2005. 2.5.4, 3.4.3

[132] David Wettergreen et al. Second experiments in the robotic investigation of life in the Atacama desert of Chile. In *8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005. 2.5.4, 3.4.3

[133] J. Y. Wong. *Theory of Ground Vehicles*. Wiley, New York, third edition, 2001. 2.3, 2.3, 2.4.3, 2.4.3

[134] J. Y. Wong and A. R. Reece. Prediction of rigid wheel performance based on the analysis of soil-wheel stresses. *Journal of Terramechanics*, 4(1):81–98, 1967. 2.4.3, C

[135] He Xu, Xing Liu, Hu Fu, Bagus Bhirawa Putra, and Long He. Visual contact angle estimation and traction control for mobile robot in rough-terrain. *Journal of Intelligent & Robotic Systems*, July 2013. 3.4.3

[136] K. Yoshida. The SpaceDyn: a MATLAB toolbox for space and mobile robots. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 1999. 2.1.2, 2.1, 2.2.1

[137] Wei Yu, Oscar Ylaya Chuy, Emmanuel G. Collins, and Patrick Hollis. Analysis and experimental verification for dynamic modeling of a skid-steered wheeled vehicle. *Transactions on Robotics*, 26(2):340–353, April 2010. 2.1.2, 2.1, 3.4.1

[138] Xiaoping Yun and Nilanjan Sarkar. Unified formulation of robotic systems with holonomic and nonholonomic constraints. *Transactions on Robotics*, 14(4):640–650, 1998. 2.4.2, 2.4.2, 2.4.2

[139] Yilin Zhao and Spencer L. Bement. Kinematics, dynamics and control of wheeled mobile robots. In *International Conference on Robotics and Automation*. IEEE, 1992. 2.1.2

Glossary

The definitions provided here are for the usage of terms *in this document only*, which may be more restrictive than general usage.

Term	Definition
actuator model	A function of the form in (2.73), which maps joint rates and commanded joint rates to output forces/torques.
coordinate system	i.e. a Cartesian coordinate system. A coordinate system is required to uniquely express position or other quantities of motion numerically. *Reference frames* have associated coordinate systems, but are a distinct concept. (Section 1.3)
differential algebraic equation (DAE)	Like ordinary differential equations (ODEs), DAEs contain a function of one independent variable and its derivatives. Unlike for ODEs the equation cannot be algebraically solved for all components of the function derivative. DAEs are differential equations with algebraic constraints. (Section 2.1). cf. *ordinary differential equation*
dynamic model	Generally speaking, a mathematical model that describes a system's behavior over time. In this document, dynamic model is often used more restrictively to mean a second-order, force-driven model that accounts for inertia. cf. *kinematic model*
formulation (model)	Expression of a model as a system of mathematical equations, and the specification of algorithms to solve those equations and predict motion. (Section 1.1.1)
holonomic (constraint)	A constraint of the form $c(\mathbf{x}, t) = 0$, i.e. that is not dependent on velocity $\dot{\mathbf{x}}$ (Section 2.1). cf. *nonholonomic*
joint space	A mathematical space whose dimension equals the number of degrees of freedom (DOF) of the WMR. The body frame is connected to the world frame by a fictitious 6-DOF joint, and all remaining frames are connected to their parents via 1-DOF joints; thus the dimension of joint space exceeds the number of WMR model frames by 5. Joint space velocity ($\dot{\mathbf{q}}'$) and acceleration ($\ddot{\mathbf{q}}'$) are defined in Section 2.2.1. cf. *state space*
kinematic model	A first-order, velocity-driven model that (typically) does not account for inertia. Our enhanced kinematic model does account for inertia to a limited extent (Section 2.3). cf. *dynamic model*.

Term	Definition
local coordinates	"in local coordinates" specifies that a quantity for an object is expressed in the coordinates of that object. For example, the velocity of frame a in local coordinates is ${}^{a}\vec{v}_{a}$ (Section 1.3).
model predictive control (MPC)	Use of a predictive motion model to determine the sequence of inputs that produce a desired output trajectory, or the closest feasible trajectory. cf. *model predictive planning*
model predictive planning (MPP)	Use of a predictive motion model to obtain feasible candidate trajectories, evaluate them according to some cost function, and select the best one for execution [71]. In receding horizon MPP the selected trajectory is only partially executed before replanning. cf. *model predictive control*
nonholonomic (constraint)	A constraint of the form $c(\dot{\mathbf{x}}, \mathbf{x}, t) = 0$, i.e. that is dependent on velocity $\dot{\mathbf{x}}$ (Section 2.1). Systems with nonholonomic constraints are nonintegrable, underactuated. cf. *holonomic*
odometry (wheel)	The dead-reckoning of WMR position from proprioceptive measurements, including wheel velocities and other joint angles/rates. GPS and IMUs are not used in wheel odometry.
Open Dynamics Engine (OpenDE)	A popular, open-source software library for rigid body dynamics simulation [116].
ordinary differential equation (ODE)	An ODE is an equation containing a function of one independent variable (e.g. time) and its derivatives (Section 2.1). cf. *differential algebraic equation*
reference frame	Physical quantities like velocity must be measured with respect to a reference frame. The reference frame is distinct from the *coordinate system* that a quantity is expressed in [63]. For example, ${}^{c}\vec{v}_{a}^{b}$ is the velocity of frame a with respect to reference frame b, expressed in the coordinate system associated with frame c (Section 2.1).
rough terrain	Terrain is rough (or uneven) if local patches the size of the WMR footprint are not well approximated by a plane. In contrast, smooth terrain can be well-approximated as locally planar. Level terrain is not sloped, i.e. the average surface normal is aligned with the gravity vector. Flat terrain is both planar and level (e.g. a parking lot or hallway).
state space	The space spanned by the state vector, defined in Section 2.2.1. The state vector comprises the pose of the body frame and joint displacements. The dimension of state space may differ from the dimension of joint space, depending on the orientation representation used. cf. *joint space*
stochastic model	Refers to the linearized error dynamics for stochastic error, expressed as a differential equation in (3.13). Calibration of the stochastic model parameters improves the estimates of covariance for motion predictions. cf. *systematic model*

Term	Definition
systematic model	Refers to the deterministic system dynamics, expressed as a differential equation in (3.1). Calibration of the systematic model parameters removes systematic errors from motion predictions. cf. *stochastic model*
wheel-ground contact model	A function of the form in (2.71) which maps slip velocities, the product of wheel radius and angular velocity, and contact height error to contact forces. See Appendix C for examples.

Appendix

A Orientation Representations

In this document, orientation can be represented by either a rotation matrix (R) or vector ($\mathbf{o}$). Our WMR model formulations use rotation matrices internally, but the state ($\mathbf{q}$) contains a vector representing body frame orientation (2.10). This vector comprises either Euler angles or a quaternion. Here we briefly explain all orientation representations and conversion between them. We also explain the conversion between angular velocity and the time derivative of the orientation vector. More details about orientation representations for WMRs is provided in [63].

Script notation for rotation matrices and orientation vectors is explained in Section 1.3. In short, recall that R_a^b or $\mathbf{o}_a^b$ denotes the orientation of frame a relative to frame b. In addition, recall that all frames use right-handed coordinate systems, and the body frame uses the coordinate system convention from the ISO 8855 standard: x-forward, y-left, z-up [54].

Rotation matrices. Rotation matrices must be 3×3 and orthonormal (or orthogonal with a determinant equal to 1). When used to express the orientation of a coordinate system, the columns are unit vectors in the directions of the x, y, and z axes. This fact is used in the wheel Jacobian calculation (Algorithm 2) to obtain the axes for revolute/prismatic joints from rotation matrices.

The following matrices represent rotations about the x, y, and z axes by the angle θ:

$$\mathrm{Rot}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\theta & -s\theta \\ 0 & s\theta & c\theta \end{bmatrix} \tag{A.1}$$

$$\mathrm{Rot}_y(\theta) = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \tag{A.2}$$

$$\mathrm{Rot}_z(\theta) = \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

where $s\theta = \sin(\theta)$ and $c\theta = \cos(\theta)$.

Rotation matrices have several important properties. First, their transpose is equivalent to their inverse:

$$R^T = R^{-1} \tag{A.4}$$

$$(R_a^b)^T = (R_a^b)^{-1} = R_b^a \tag{A.5}$$

Second, the relative orientation of two frames in a chain can be computed from the product of rotation matrices for all frames between them:

$$R_a^c = R_b^c R_a^b \tag{A.6}$$
$$R_a^d = R_c^d R_b^c R_a^b \tag{A.7}$$
$$R_a^e = R_d^e R_c^d R_b^c R_a^b \tag{A.8}$$
$$\vdots$$

Multiplication is *not commutative* for products of more than two rotation matrices.

Euler angles. Euler angles represent a sequence of three *elemental rotations*, i.e. about the axes of a coordinate system. Many Euler angle conventions are possible; they differ by which axes are rotated about and in what order. We denote the three Euler angles as roll (ϕ), pitch (θ), and yaw (ψ) and define them as sequential rotations about the x, y, and z axes. Thus, the conversion from Euler angles to a rotation matrix is:

$$R = \mathrm{Rot}_z(\psi)\,\mathrm{Rot}_y(\theta)\,\mathrm{Rot}_x(\phi) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - s\phi c\psi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{A.9}$$

(A.9) can be solved for the opposite conversion from a rotation matrix to Euler angles. The following are computed sequentially:

$$\psi = \mathrm{atan2}(R(2,1), R(1,1)) \tag{A.10}$$
$$\phi = \mathrm{atan2}(R(3,2), R(3,3)) \tag{A.11}$$
$$c\theta = \frac{R(3,3)}{c\phi} = \frac{R(3,2)}{s\phi} \tag{A.12}$$
$$\theta = \mathrm{atan2}(-R(3,1), c\theta) \tag{A.13}$$

The function $\mathrm{atan2}()$ returns values in the range of $-\pi$ to π. Either expression for $c\theta$ may be used to avoid a singularity.

The matrix V in (2.14) requires the submatrix (Ω) to convert from angular velocity (in body coordinates) to Euler angle rates. This conversion is derived in [63]; we state it here for convenience.

$$\dot{\mathbf{o}}_b^w = \Omega({}^b\vec{\omega}_b^w) \tag{A.14}$$
$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{A.15}$$

where $t\theta$ is short for $\tan(\theta)$.

Notice there is a singularity when the pitch (θ) equals $\pm\pi/2$. Nearing this singularity is commonly called "gimbal lock." The Euler angle rates ($\dot{\mathbf{o}}$) represent rotations about three axes, but not orthogonal axes. Gimbal lock occurs when two of those axes become nearly aligned (in

this case the roll and yaw axes). In practice this is seldom a problem for WMRs; their pitch angles never approach $\pm\pi/2$ so long as they stay in contact with the terrain.

Euler angles may be preferable to quaternions in some cases because they are human-readable, and they clearly distinguish between attitude (i.e. roll/pitch) and yaw. When initializing terrain contact in a simulation (Section 2.2.4) one often lets attitude vary while keeping yaw fixed. Similarly, in odometry one might provide roll/pitch measurements but not yaw measurements as inputs. Though many use "yaw" and "heading" interchangeably, they are distinguished as follows: yaw describes the orientation of the vehicle whereas heading describes the direction of linear velocity.

Quaternions. Quaternions express orientation as a vector with four elements. They are less human-readable than Euler angles, but do not suffer from gimbal lock.

Quaternions are similar to an axis and angle representation of orientation:

$$\mathbf{q} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2})\hat{u} \end{bmatrix} \tag{A.16}$$

where $\hat{u}$ is the axis of rotation and θ is the angle.

When using quaternions instead of Euler angles, equations for the following conversions are required: from quaternion to rotation matrix, from rotation matrix to quaternion, and from angular velocity to the time derivative of the quaternion. These can be found in other references, such as [85].

B Spatial Vector Algebra

Spatial Vector Algebra (SVA) was developed by Featherstone [36] for the concise expression of rigid body dynamics equations. Some description of SVA notation is provided in Section 1.3, but a more detailed explanation is provided here.

Spatial vectors. Spatial vectors comprise an angular and linear component, each represented by a Cartesian coordinate vector (making 6 elements total). Spatial vectors can represent motion (i.e. velocity, acceleration) or force:

$$\vec{v} = \begin{bmatrix} \vec{\omega} \\ \vec{v} \end{bmatrix}, \quad \vec{a} = \begin{bmatrix} \vec{\alpha} \\ \vec{a} \end{bmatrix}, \quad \vec{f} = \begin{bmatrix} \vec{\tau} \\ \vec{f} \end{bmatrix} \tag{B.1}$$

Plücker transforms. To change the coordinates that spatial vectors are expressed in, one multiplies by Plücker transforms. Like homogeneous transforms, they represent both rotation and translation:

$$X_a^b = \begin{bmatrix} R_a^b & [0] \\ [0] & R_a^b \end{bmatrix} \begin{bmatrix} I & [0] \\ [{}^a\vec{r}_b]_\times^T & I \end{bmatrix} = \begin{bmatrix} R_a^b & [0] \\ R_a^b[{}^a\vec{r}_b]_\times^T & R_a^b \end{bmatrix} \tag{B.2}$$

An equivalent homogeneous transform to the Plücker transform in (B.2) is:

$$T_a^b = \begin{bmatrix} R_a^b & {}^b\vec{r}_a \\ [0] & 1 \end{bmatrix} \tag{B.3}$$

Thus one can convert a homogeneous transform to a Plücker transform as follows. Blocks (1,1) and (2,2) of the Plücker transform equal the rotation matrix in the homogeneous transform:

$$X_a^b([1],[1]) = X_a^b([2],[2]) = R_a^b \tag{B.4}$$

$$= T_a^b([1],[1]) \tag{B.5}$$

$$\tag{B.6}$$

Block (2,1) of the Plücker transform can be computed from the translation vector in the homogenous transform as follows:

$$X_a^b([2],[1]) = R_a^b[{}^a\vec{r}_b]_\times^T \tag{B.7}$$

$$= -R_a^b[{}^a\vec{r}_b]_\times \tag{B.8}$$

$$= [\vec{t}]_\times R_a^b \tag{B.9}$$

where:

$$\vec{t} = -R_a^b({}^a\vec{r}_b) = T_a^b([1],[2]) \tag{B.10}$$

Our script notation denotes a Plücker transform's effect on motion vectors. For example, the coordinates of spatial velocity are transformed as follows:

$${}^b\vec{v} = X_a^b({}^a\vec{v}) \tag{B.11}$$

The equivalent Plücker transform for force vectors is the *inverse transpose* of the Plücker transform for motion vectors:

$$^{b}\vec{f} = (X_a^b)^{-T}(^{a}\vec{f}) \tag{B.12}$$

Spatial inertia. Spatial inertia is a symmetric, 6×6 matrix that represents the mass (m), center of mass ($\vec{r}_{cm}$), and moment of inertia about the center of mass (I_{cm}) of a rigid body. Spatial inertia is computed as follows:

$$I_{spatial} = \begin{bmatrix} I_{cm} + m(CC^T) & mC \\ mC^T & mI_3 \end{bmatrix} \tag{B.13}$$

where $C = [\vec{r}_{cm}]_\times$ and I_3 is a 3×3 identity matrix. To transform spatial inertia, one pre-multiplies by a Plücker transform (for force) and post-multiplies by its transpose. For an example, see the Composite Rigid Body Algorithm (Algorithm 3).

Spatial cross products. SVA equations sometimes require the cross product of spatial velocity with another vector. The operation differs depending on whether the other vector represents motion or force.

The cross product of velocity with a motion vector ($\vec{m}$) is defined as:

$$\vec{v} \times_m \vec{m} = \begin{bmatrix} [\vec{\omega}]_\times & [0] \\ [\vec{v}]_\times & [\vec{\omega}]_\times \end{bmatrix} \vec{m} \tag{B.14}$$

The cross product of velocity with a force vector ($\vec{f}$) is defined as:

$$\vec{v} \times_f \vec{f} = \begin{bmatrix} [\vec{\omega}]_\times & [0] \\ [\vec{v}]_\times & [\vec{\omega}]_\times \end{bmatrix}^T \vec{f} \tag{B.15}$$

Note that the block matrix in (B.14) is *transposed* relative to the block matrix in (B.15).

C Wheel-Ground Contact Models

As explained in Section 2.4.3, our dynamics formulation supports any wheel-ground contact model expressed as a function of the form:

$$\vec{f}_c = \vec{f}(\vec{v}_c, R\omega, \Delta z) \tag{C.1}$$

where $\vec{f}_c$ is the force at the contact point and $\vec{v}_c$ is the contact point velocity (relative to the ground), both expressed in local coordinates. These each have three elements:

$$\vec{f}_c = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \quad \vec{v}_c = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{C.2}$$

The remaining inputs in (C.1) are: the product of wheel radius and angular velocity $R\omega$, and the contact height error Δz, explained in Section 2.2.1.

Here we provide equations for the three wheel-ground contact model types used in our benchmark tests (Section 2.5.4): piecewise-linear, Pacejka/Nicolas-Comstock, and Ishigami terramechanics-based.

Many automotive and robotics publications provide wheel-terrain interaction models like these, but typically only consider a single wheel. Our WMR dynamics formulation helps to utilize this related work; with just a few lines of code one can relate wheel-terrain interaction models to full WMR mobility.

Piecewise-linear. The piecewise-linear wheel-ground contact model replicates Open Dynamics Engine's Coulomb friction approximation for contact forces.

Normal force is computed via a spring/damper equation, but only if contact height error is less than zero (indicating penetration).

$$f'_z = \begin{cases} K_p \Delta z + K_d v_z & \text{if } \Delta z < 0 \\ 0 & \text{otherwise} \end{cases} \tag{C.3}$$

Normal force cannot be negative (i.e. the ground cannot pull on the wheel), so it is bounded at zero if necessary:

$$f_z = \begin{cases} f'_z & \text{if } f'_z \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{C.4}$$

Open Dynamics Engine computes normal force similarly with erp/cfm parameters, and the same complementarity condition and lower bound.

Longitudinal and lateral force are then computed as follows:

$$f'_x = C_x v_x + C_{rr} R\omega \tag{C.5}$$

$$f'_y = C_y v_y \tag{C.6}$$

The C_x and C_y coefficients correspond to the "force-dependent slip" parameters in Open Dynamics Engine; they specify a component of force that is linearly proportional to slip velocity.

The $C_{rr}R\omega$ term accounts for rolling resistance; it prevents the vehicle from coasting at constant speed due to loss of energy.

Like Open Dynamics Engine, we also limit tangential contact forces:

$$f_x = \begin{cases} f'_x & \text{if } f'_x \leq \mu_x f_z \\ \mu_x f_z & \text{otherwise} \end{cases} \tag{C.7}$$

$$f_y = \begin{cases} f'_y & \text{if } f'_y \leq \mu_y f_z \\ \mu_y f_z & \text{otherwise} \end{cases} \tag{C.8}$$

This is a "pyramid" approximation to a Coulomb friction cone; tangential force is bounded independently in two orthogonal directions, proportional to normal force. A true Coulomb friction limit bounds the norm of the tangential force vector:

$$\left\| \begin{bmatrix} f_x \\ f_y \end{bmatrix} \right\| \leq \mu f_z \tag{C.9}$$

If desired this true Coulomb friction limit could be enforced via force-balance optimization, though it would introduce a nonlinear dependency between forces.

Figure C.1 shows a plot of normalized force vs. slip velocity for the piecewise-linear wheel-ground contact model.

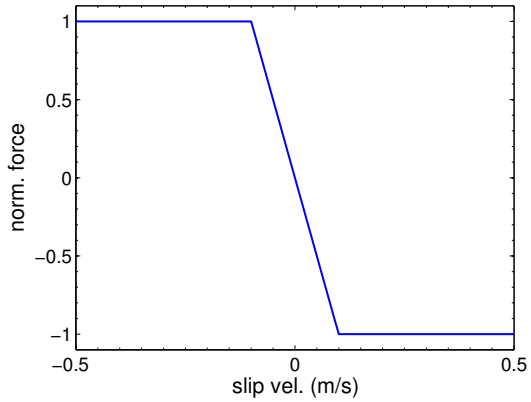


Figure C.1: Normalized force (f_x/f_z or f_y/f_z) vs. slip velocity (v_x or v_y) for the piecewise-linear wheel-ground contact model. Friction resists motion, so force and slip velocity have opposite signs. This plot assumes no rolling resistance ($C_{rr} = 0$).

Pacejka/Nicolas-Comstock. The Pacejka/Nicolas-Comstock wheel-ground contact model is an empirical model for pneumatic tires. We use the version presented in [19][18].

Normal force is computed exactly as in the piecewise-linear model, according to (C.3) and (C.4).

Most tire models parametrize force not directly over slip velocities, but over slip ratio (s) and

angle (α). The definition of these terms varies, but for the Pacejka model they are defined as:

$$s = -\frac{v_x}{V_x} \tag{C.10}$$

$$\alpha = \tan^{-1}\left(\frac{v_y}{V_x}\right) \tag{C.11}$$

where V_x is the longitudinal velocity of the wheel center:

$$V_x = v_x + R\omega \tag{C.12}$$

According to these definitions slip angle is bounded between $-\pi/2$ and $\pi/2$, but slip ratio is unbounded. Both are poorly defined near the singularity $V_x = 0$, i.e. when the vehicle is near stationary.

The Pacejka "magic formulas" for normalized longitudinal and lateral force are:

$$f'_x(s)/f_z = D_s \sin(C_s \tan^{-1}(B_s(1-E_s)K_s s + E_s \tan^{-1}(B_s K_s s))) \tag{C.13}$$

$$f'_y(\alpha)/f_z = D_a \sin\left(C_a \tan^{-1}\left(B_a(1-E_a)K_a\frac{2\alpha}{\pi} + E_a \tan^{-1}\left(B_a K_a \frac{2\alpha}{\pi}\right)\right)\right) \tag{C.14}$$

$$\tag{C.15}$$

Typical values for the coefficients are provided in Table C.1. Figure C.2 plots both f'_x as a function of s, and f'_y as a function of α. Notice how, above a critical value, the magnitude of force *decreases* with slip.

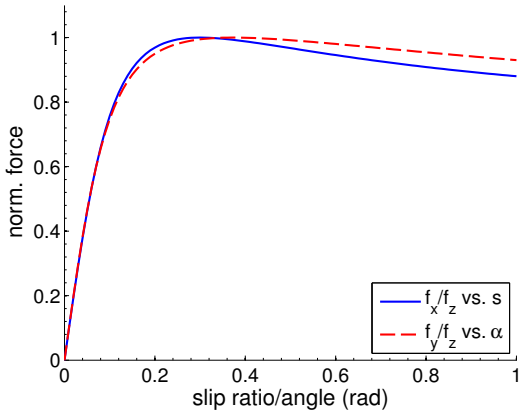


Figure C.2: Normalized force vs. slip ratio/angle for the Pacejka wheel-ground contact model *without* the Nicolas-Comstock equation

Of course, longitudinal force is not actually independent of slip angle, neither is lateral force independent of slip ratio. We couple them using the Nicolas-Comstock equation as follows:

$$f_x(s,\alpha)/f_z = \operatorname{sgn}(V_x)\mu F \cdot s \tag{C.16}$$

$$f_y(s,\alpha)/f_z = -\operatorname{sgn}(V_x)\mu F \cdot \tan(\alpha) \tag{C.17}$$

$$F = \frac{f'_x f'_y}{\sqrt{s^2(f'_y)^2 + (f'_x)^2 \tan^2(\alpha)}} \tag{C.18}$$

Table C.1: Parameter values for Pacejka/Nicolas-Comstock model

B_s	1/15	B_a	8/75
C_s	1.5	C_a	1.5
D_s	1.0	D_a	1.0
E_s	0.3	E_a	0.6
K_s	100.0	K_a	100.0

The denominator of F equals zero when either slip ratio or angle is zero (causing either f'_y or f'_x to also be zero). To avoid this singularity, thresholding of s and α away from zero may be necessary.

The $\mathrm{sgn}(V_x)$ term is not included in Brach and Brach [19], but it is necessary to ensure the correct sign of f_x and f_y when the vehicle can drive in reverse. The coefficient of friction was also added to simulate low traction surfaces ($\mu < 1$). If desired, a rolling resistance term can also be added to longitudinal force equation, as in (C.5).

Recall that force-balance optimization requires the derivative of contact force ($\vec{f}_c$) with respect to the inputs ($\vec{v}_c, R\omega, \Delta z$). Deriving this Jacobian analytically can be difficult for such complicated expressions; we did so using MATLAB's symbolic math toolbox.

Plots of normalized longitudinal and lateral force vs. slip ratio and angle for the Pacejka/Nicolas-Comstock wheel-ground contact model are shown in Figure C.3. Unlike for the piecewise-linear model, each force depends nonlinearly on both slip ratio and angle.

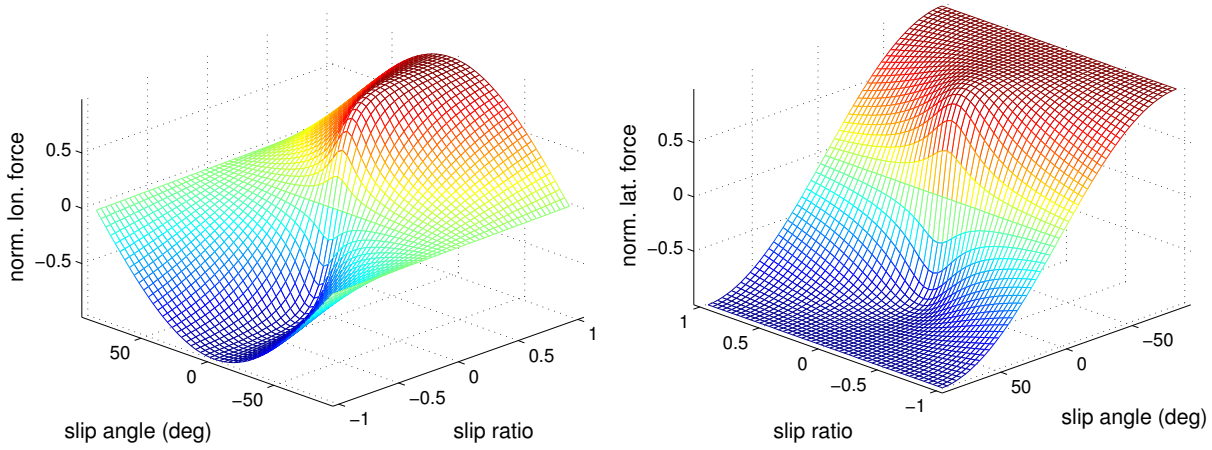


Figure C.3: (Left) Normalized longitudinal force (f_x/f_z) vs. slip ratio and angle for the Pacejka/Nicolas-Comstock wheel-ground contact model. (Right) Normalized lateral force (f_y/f_z) vs. slip ratio and angle.

Ishigami terramechanics-based. There are many terramechanics-based models in the robotics literature for rigid wheels in loose soil. Most of these models are based on early work by Bekker [11] and Wong and Reece [134]. These models are computationally-expensive, as they require the integration of stress distributions along the wheel surface. We use the terramechanics-based model from Ishigami et al. [52], which unlike some publications considers lateral as well as longitudinal forces.

Table C.2: Parameters for the Ishigami terramechanics-based model and sample values from Ishigami et al. [52]

param.	value	unit	description
c	0.80	kPa	cohesion stress
ϕ	37.2	deg	friction angle
X_c	26.4	deg	soil distractive angle
k_c	1.37×10^3	N/m^{n+1}	pressure-sinkage module
k_ϕ	8.14×10^5	N/m^{n+2}	pressure-sinkage module
n	1		sinkage exponent
a_0	0.40		(determines θ_m)
a_1	0.15		
ρ_d	1.6×10^3	kg/m^3	soil density
λ	0.9-1.1		wheel sinkage ratio
k_x	$0.043\beta + 0.036$	m	soil deformation module
k_y	$0.020\beta + 0.013$	m	soil deformation module

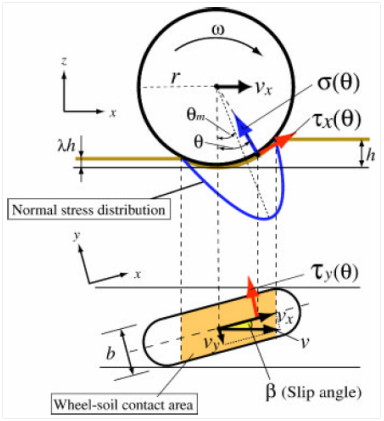


Figure C.4: Diagram of stress distributions on the wheel surface. Reproduced from Ishigami et al. [52].

Ishigami's model also requires slip ratio and angle, but defines slip ratio differently than in the Pacejka model, and denotes slip angle by β instead of α:

$$s = -\frac{v_x}{R\omega} \tag{C.19}$$

$$\beta = \tan^{-1}\left(\frac{v_y}{V_x}\right) \tag{C.20}$$

This definition of slip ratio is also unbounded, and singular when the wheel is stationary ($\omega = 0$).

In loose soil, contact does not occur at a point but over a region specified by entry (θ_f) and exit (θ_r) angles:

$$\theta_f = \cos^{-1}(1 - h/R) \tag{C.21}$$

$$\theta_r = \cos^{-1}(1 - \lambda h/R) \tag{C.22}$$

where $h = -\Delta z$ (the negative of contact height error, or sinkage) and R is wheel radius. λ is one of the many soil parameters required for this model; calibrated parameter values from [52] are provided in Table C.2.

The normal stress varies along the wheel surface as a function of the angle θ from θ_r to θ_f (Figure C.4). The peak stress occurs at θ_m, which is a function of slip ratio:

$$\theta_m = (a_0 + a_1 s)\theta_f \tag{C.23}$$

The equation for normal stress as function of the angle θ is:

$$\sigma(\theta) = \begin{cases} r^n \left(\frac{k_c}{b} + k_\phi\right) (\cos\theta - \cos\theta_f)^n & \theta_m \le \theta < \theta_f \\ r^n \left(\frac{k_c}{b} + k_\phi\right) \left(\cos\left(\theta_f - \frac{\theta - \theta_r}{\theta_m - \theta_r}(\theta_f - \theta_m)\right) - \cos\theta_f\right)^n & \theta_r < \theta < \theta_m \end{cases} \tag{C.24}$$

In (C.24), b is the wheel width, and r, n, k_c, and k_ϕ are soil properties defined in Table C.2.

One must also compute shear stress distributions in the longitudinal (x) and lateral (y) directions:

$$\tau_x(\theta) = (c + \sigma(\theta)\tan\phi)(1 - e^{-j_x(\theta)/k_x}) \tag{C.25}$$

$$\tau_y(\theta) = (c + \sigma(\theta)\tan\phi)(1 - e^{-j_y(\theta)/k_y}) \tag{C.26}$$

$$\tag{C.27}$$

where j_x and j_y are soil deformations, computed as follows:

$$j_x = r(\theta_f - \theta - (1 - s)(\sin\theta_f - \sin\theta)) \tag{C.28}$$

$$j_y = r(1 - s)(\theta_f - \theta)\tan\beta \tag{C.29}$$

These are the equations in [52], but the signs of τ_x and τ_y may need to be changed, such that the shear stress direction opposes the slip direction. An alternative calculation for shear stress, that more realistically assumes *isotropic* soil properties, is provided by Jia et al. [56].

The net contact forces on the wheels are computed by integrating these stress distributions:

$$f_x = rb\int_{\theta_r}^{\theta_f} (\tau_x(\theta)\cos\theta - \sigma(\theta)\sin\theta)d\theta \tag{C.30}$$

$$f_y = \int_{\theta_r}^{\theta_f} (rb\cdot\tau_y(\theta) + R_b(r - h(\theta)\cos\theta))\,d\theta \tag{C.31}$$

$$f_z = rb\int_{\theta_r}^{\theta_f} (\tau_x(\theta)\sin\theta + \sigma(\theta)\cos\theta)d\theta \tag{C.32}$$

$$\tag{C.33}$$

The R_b term inside the integral for lateral force (f_y) accounts for the bulldozing of soil by the wheel's sidewall. R_b can be computed as a function of penetration height along the wheel surface $h(\theta)$ and soil parameters:

$$R_b(h) = D_1\left(c\cdot h(\theta) + D_2\frac{\rho_d\cdot h^2(\theta)}{2}\right) \tag{C.34}$$

$$D_1 = \cot X_c + \tan(X_c + \phi) \tag{C.35}$$

$$D_2 = \cot X_c + \frac{\cot^2 X_c}{\cot\phi} \tag{C.36}$$

According to these equations, the bulldozing resistance per unit width (R_b) scales quadratically with sinkage, $h(\theta)$.

Numerical integration of the stress distributions can be very costly. If parameter values are fixed, we can address this by precomputing a lookup table of contact forces ($\vec{f}_c$) for ranges of s, β, and Δz values. Then during simulation we can quickly compute contact forces by trilinear interpolation between values in the table. We can compute the derivative required for force-balance optimization using finite-differences. A lookup table is not possible when parameter values are being continually updated via online calibration. Terramechanics-based models have the disadvantage of numerous parameters, making calibration expensive and more likely to converge to local minima.

Plots of contact forces vs. slip ratio and angle for the Ishigami wheel-ground contact model are shown in Figure C.5. These plots assume a fixed value for Δz. The discontinuity in lateral force around $\beta = 0$ is due to the bulldozing force. The decrease in normal force with increased slip ratio matches an experimentally observed phenomenon; the more wheels slip the deeper they sink into loose soil. This dependence of normal force on slip is not present in the Pacejka or piecewise-linear models.

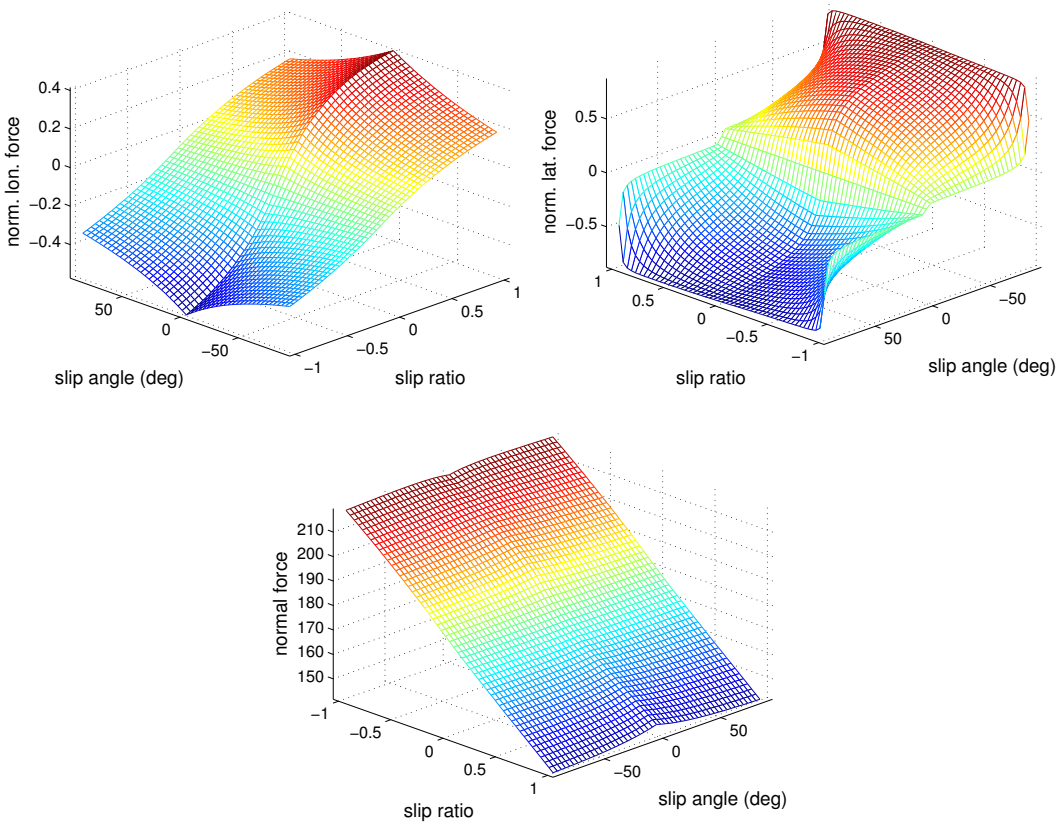


Figure C.5: (Top Left) Normalized longitudinal force (f_x/f_z) vs. slip ratio and angle for the Ishigami terramechanics-based wheel-ground contact model. (Top Right) Normalized lateral force (f_y/f_z) vs. slip ratio and angle. (Bottom) Normal force (f_z) vs. slip ratio and angle. Assumes constant Δz, and uses wheel radius/width from [52].

D Line Search Algorithm

In force-balance optimization, we minimize a cost function using Newton's method combined with a line search algorithm (Section 2.4.3). The line search algorithm determines the step size α in (2.84), such that the strong Wolfe conditions are satisfied.

A line search is the minimization of a univariate function $\phi(\alpha) = f(\mathbf{x} + \alpha\mathbf{p})$ where $\mathbf{p}$ is the the search direction. In our case the search direction is determined by Newton's method:

$$\mathbf{p} = -[Hf(\mathbf{x})]^{-1}\nabla f(\mathbf{x}) \tag{D.1}$$

Often ϕ need not be minimized *exactly*, but only *sufficiently*. The Wolfe conditions determine whether or not sufficient minimization has been achieved:

$$f(\mathbf{x} + \alpha\mathbf{p}) \leq f(\mathbf{x}) + c_1\alpha\mathbf{p}^T\nabla f(\mathbf{x}) \tag{D.2}$$

$$\mathbf{p}^T\nabla f(\mathbf{x} + \alpha\mathbf{p}) \geq c_2\mathbf{p}^T\nabla f(\mathbf{x}) \tag{D.3}$$

Example values for the coefficients are $c_1 = 10^{-4}$ and $c_2 = 0.9$. The first condition ensures that f has decreased sufficiently. The second condition, called the "curvature condition," ensures that slope has decreased sufficiently.

The *strong* Wolfe conditions enforce a modified curvature condition:

$$\left|\mathbf{p}^T\nabla f(\mathbf{x} + \alpha\mathbf{p})\right| \leq c_2\left|\mathbf{p}^T\nabla f(\mathbf{x})\right| \tag{D.4}$$

This ensures that α is near a critical point of ϕ.

A step size α that satisfies the strong Wolfe conditions can be found using the line search method published by Nocedal and Wright [92]. First, Nocedal and Wright's Algorithm 3.2 checks if a full step ($\alpha = 1$) is acceptable, i.e. satisfies the strong Wolfe conditions. If not, it checks if there must exist an acceptable step size in the interval 0 to 1. If so, Nocedal and Wright's Algorithm 3.3 (the "zoom" function) then uses bisection to find an acceptable value in the interval.

E Model Information for Test Vehicles

This section provides model information for vehicles used in tests/experiments in this thesis: the LandTamer, Zoë rover, Crusher, and RecBot. For each vehicle we provide a table of frame information (required to construct the kinematic tree), a frame diagram, and a description of any holonomic joint constraints.

Model information for the Rocky 7 rover is provided in Section 2.2.1. The columns of the frame information tables are also explained there.

The Rocky 7 rover had one additional holonomic joint constraint (Section 2.2.3). Crusher has six holonomic joint constraints (one for each "link" joint) and RecBot has two (one for each "susp" joint). Each constrains the joint displacement to equal a desired value. For example, the constraint for Crusher's linkFL joint is:

$$c = \mathbf{q}(\text{linkFL}) - \tilde{\theta} = 0 \tag{E.1}$$

where $\tilde{\theta}$ is the natural displacement. The derivative A_{joint} is simply a row vector of zeros, except for a one at the index of linkFL in joint space:

$$A_{joint} = [[0\ 0\ 0\ 0\ 0\ 0]\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \tag{E.2}$$

The index of linkFL in the frame list is 2, thus its index in joint space is 7.

The model (2.74) required for force-balance optimization is a spring-damper system. The spring stiffness is chosen to be moderate to keep wheels in contact with the ground on uneven terrain.

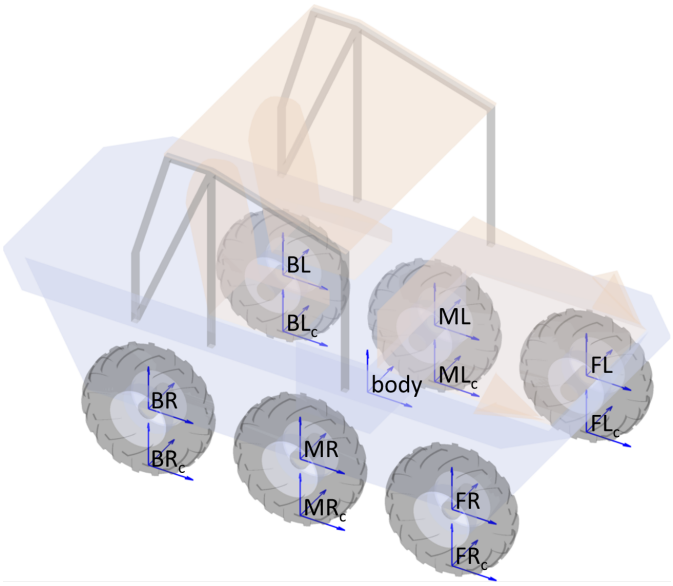


Figure E.1: LandTamer frame diagram.

Table E.1: LandTamer frame information

i	Frame	Parent	Type	Act.	x	y	z	θ_x	θ_y	θ_z
1	body (b)	world (w)								
2	**FL**	body	RY	Y	l	w	0	0	0	0
3	**FR**	body	RY	Y	l	-w	0	0	0	0
4	**ML**	body	RY	Y	0	w	0	0	0	0
5	**MR**	body	RY	Y	0	-w	0	0	0	0
6	**BL**	body	RY	Y	-l	w	0	0	0	0
7	**BR**	body	RY	Y	-l	-w	0	0	0	0

in inches: l=42, w=32.25, wheel radius=16.5, total mass = 3225 lbm

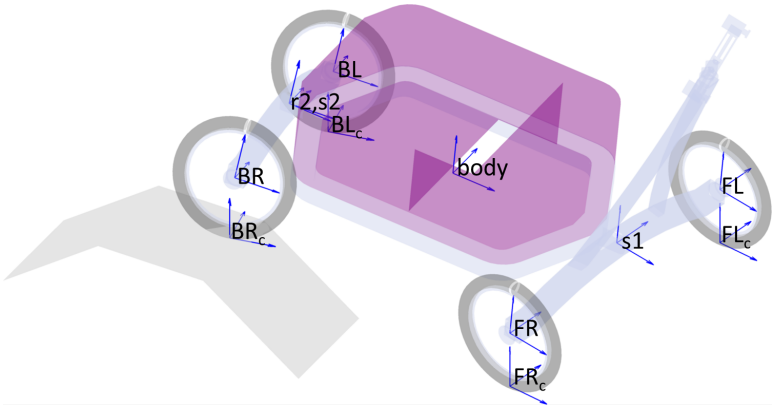


Figure E.2: Zoë rover frame diagram.

Table E.2: Zoë rover frame information

i	Frame	Parent	Type	Act.	x	y	z	θ_x	θ_y	θ_z
1	body (b)	world (w)								
2	s1	body	RZ	N	l	0	0	0	0	0
3	**FL**	s1	RY	Y	0	w	-d	0	0	0
4	**FR**	s1	RY	Y	0	-w	-d	0	0	0
5	r2	body	RX	N	-l	0	0	0	0	0
6	s2	r2	RZ	N	0	0	0	0	0	0
7	**BL**	s2	RY	Y	0	w	-d	0	0	0
8	**BR**	s2	RY	Y	0	-w	-d	0	0	0

in meters: l=.955, w=.820, d=.119, wheel radius=.325, total mass = 198 kg

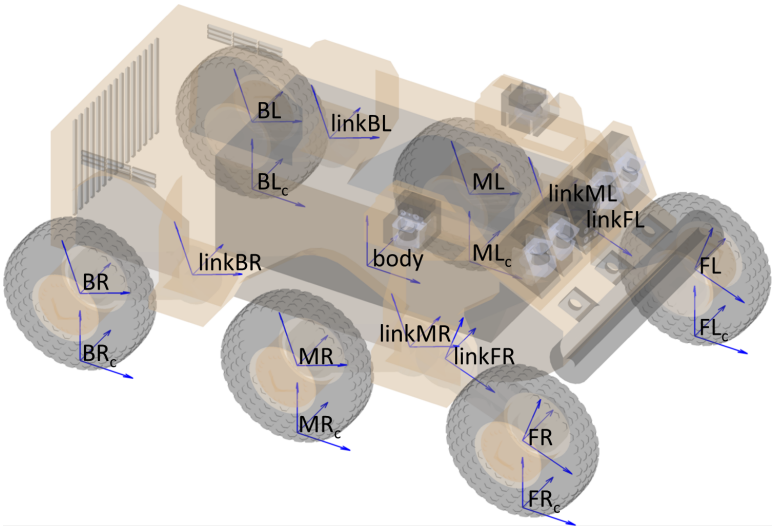


Figure E.3: Crusher frame diagram.

Table E.3: Crusher frame information

i	Frame	Parent	Type	Act.	x	y	z	θ_x	θ_y	θ_z
1	body (b)	world (w)								
2	linkFL	body	RY	N	t1	t2	t3	0	0	0
3	linkFR	body	RY	N	t1	-t2	t3	0	0	0
4	linkML	body	RY	N	t6	t2	t3	0	0	0
5	linkMR	body	RY	N	t6	-t2	t3	0	0	0
6	linkBL	body	RY	N	t7	t2	t3	0	0	0
7	linkBR	body	RY	N	t7	-t2	t3	0	0	0
8	**FL**	linkFL	RY	Y	t4	t5	0	0	0	0
9	**FR**	linkFR	RY	Y	t4	-t5	0	0	0	0
10	**ML**	linkML	RY	Y	-t4	t5	0	0	0	0
11	**MR**	linkMR	RY	Y	-t4	-t5	0	0	0	0
12	**BL**	linkBL	RY	Y	-t4	t5	0	0	0	0
13	**BR**	linkBR	RY	Y	-t4	-t5	0	0	0	0

in meters: t1=1.450, t2=1.175, t3=0.250, t4=1.000, t5=0.300, t6=1.100, t7=-1.050, wheel radius=0.700, total mass = 13200 lbm

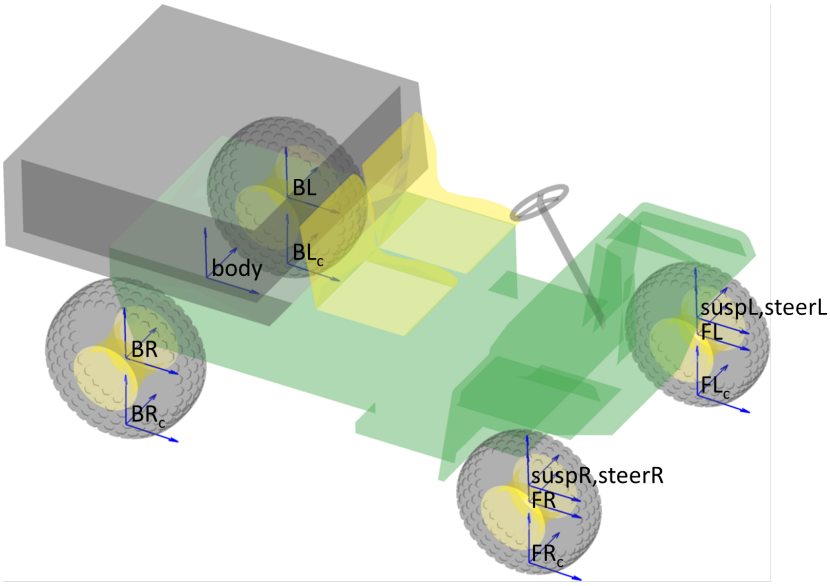


Figure E.4: RecBot frame diagram.

Table E.4: RecBot frame information

i	Frame	Parent	Type	Act.	x	y	z	θ_x	θ_y	θ_z
1	body (b)	world (w)								
2	suspL	body	PZ	N	l	w1	0	0	0	0
3	suspR	body	PZ	N	l	-w1	0	0	0	0
4	steerL	suspL	RZ	Y	0	0	0	0	0	0
5	steerR	suspR	RZ	Y	0	0	0	0	0	0
6	**FL**	steerL	RY	N	0	0	h	0	0	0
7	**FR**	steerR	RY	N	0	0	h	0	0	0
8	**BL**	body	RY	Y	0	w2	0	0	0	0
9	**BR**	body	RY	Y	0	-w2	0	0	0	0

in meters: l=1.770, w1=0.635, w2=0.610, h=-0.075, total mass = 667 kg
in inches: wheel radius=11 (FL,FR), 12 (BL,BR)

F Additional Figures

This section contains additional figures for Sections 2.5.2 and 3.4.1

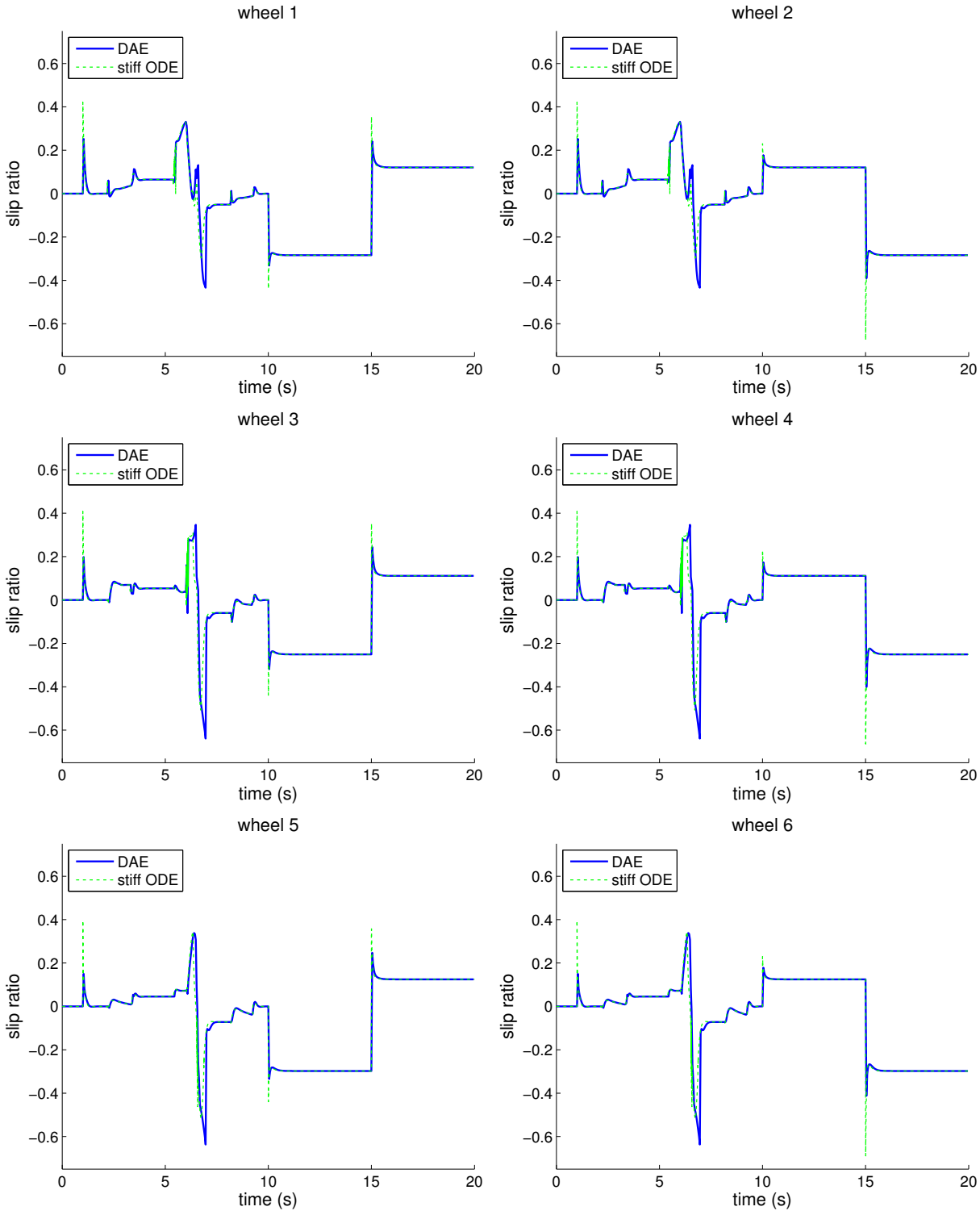


Figure F.1: Slip ratio vs. time for the LandTamer's six wheels during a dynamic simulation. Solid lines represent DAE method output, dashed lines represent stiff ODE method output. (Section 2.5.2)

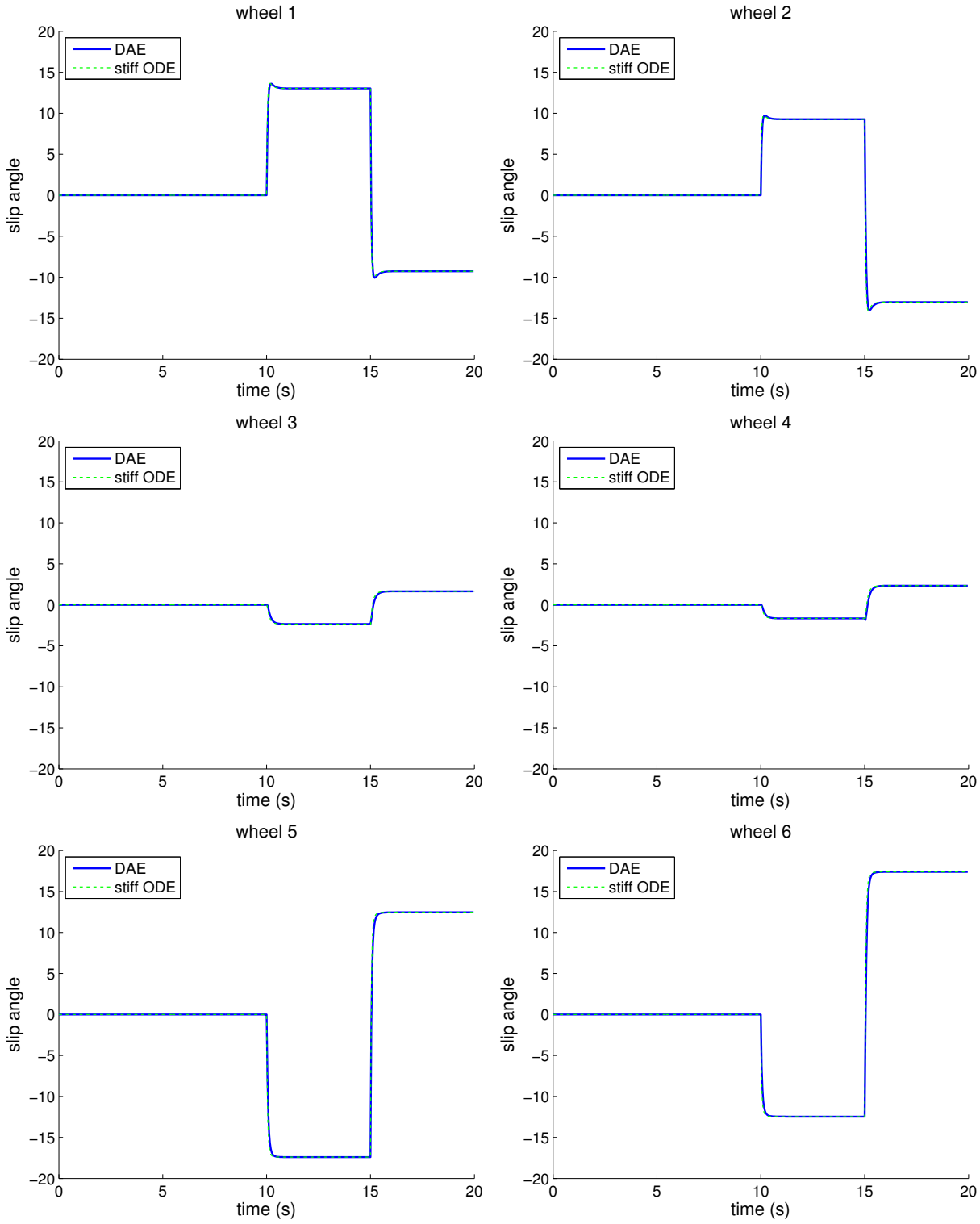


Figure F.2: Slip angle vs. time for the LandTamer's six wheels during a dynamic simulation. (Section 2.5.2)

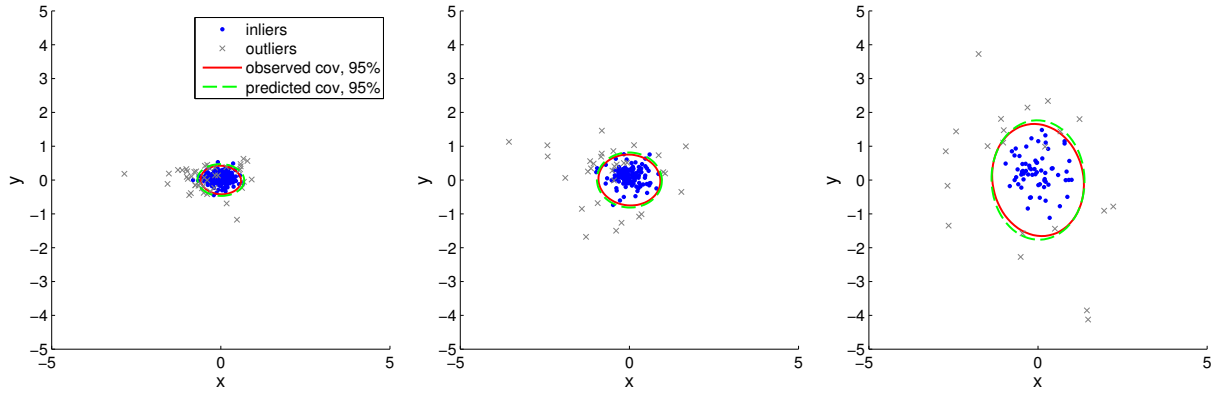


Figure F.3: Position error scatter plots for 1, 2, and 4 s intervals for Crusher at Camp Roberts. Equivalent to the plots in Figure 3.8 but with wider axes.

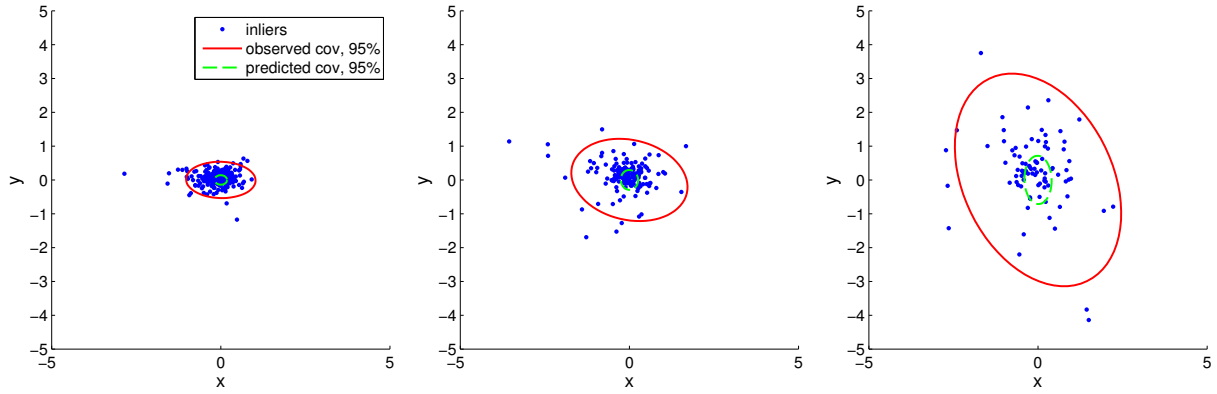


Figure F.4: Position error scatter plots for Crusher with the stochastic parameters calibrated to *velocity residuals*. The average of covariances predicted by the stochastic model (dashed green ellipses) are much smaller than the observed scatter of the errors (solid red ellipses), due to violation of the white noise assumption.

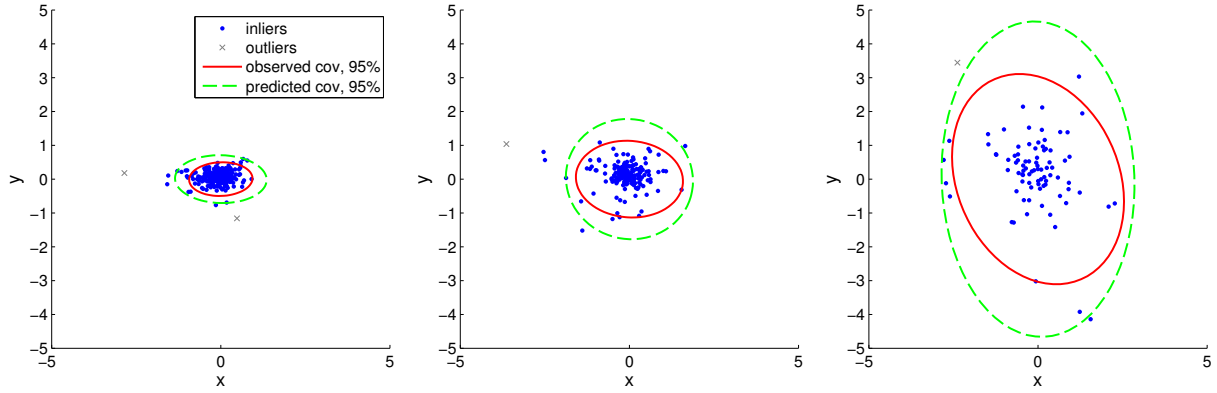


Figure F.5: Position error scatter plots for Crusher with a higher Mahalanobis distance threshold (1000) and thus fewer outliers than in Figure F.3. The covariances predicted by the stochastic model are overly large due to violation of the Gaussian noise assumption.