Traction Control Design and Integration Onboard the Mars Science Laboratory Curiosity Rover

Olivier Toupet, Jeffrey Biesiadecki, Arturo Rankin, Amanda Steffy, Gareth Meirion-Griffith, Dan Levine, Maximilian Schadegg, Mark Maimone
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-4605
olivier.toupet@jpl.nasa.gov

Abstract—**The Mars Science Laboratory (MSL) Curiosity rover experienced increasing wheel damage beginning in October 2013. While the wheels were designed to operate with considerable damage, the rate at which damage was occurring was unexpected and raised concerns regarding wheel lifetime. The Jet Propulsion Laboratory (JPL) has now developed and deployed new software on Curiosity that reduces the forces acting on the wheels. Our new Traction Control algorithm adapts each wheel's speed to fit the terrain it drives over. It does not rely on any a priori knowledge of the terrain, and instead leverages the rover's measured attitude rates and suspension angles, together with a rigid-body kinematics model, to estimate the real-time wheel-terrain contact angles and ideal, no-slip wheel angular rates. In addition, free-floating "wheelies" are detected and autonomously corrected. In this paper, we describe the algorithm, its ground testing campaign and associated challenges, and finally its validation and performance in flight. Ground test data demonstrates reductions in the forces acting on the wheels and validates the wheelie-damping capability. Secondary benefits in some terrains include a reduction in heading deviations while climbing rocks, with a reduction in slip in certain sandy terrains. Preliminary validation from flight data confirms these findings.**

TABLE OF CONTENTS

1. INTRODUCTION

In October 2013, images taken with the Mars Hand Lens Imager (MAHLI) of the Mars Science Laboratory (MSL) Curiosity Rover revealed that damage on the rover's wheels had progressed at an unexpectedly high rate. While wheel damage was expected over the course of the rover's mission, the wheel punctures seen in the images were unlike the stress concentration cracking seen after extensive high-load driving in ground testing. After a detailed investigation into the causes of wheel damage [1], the MSL project began efforts to reduce further wheel damage by altering the way the vehicle drives over obstacles. The result of these efforts is the Traction Control (TRCTL) algorithm described in this paper.

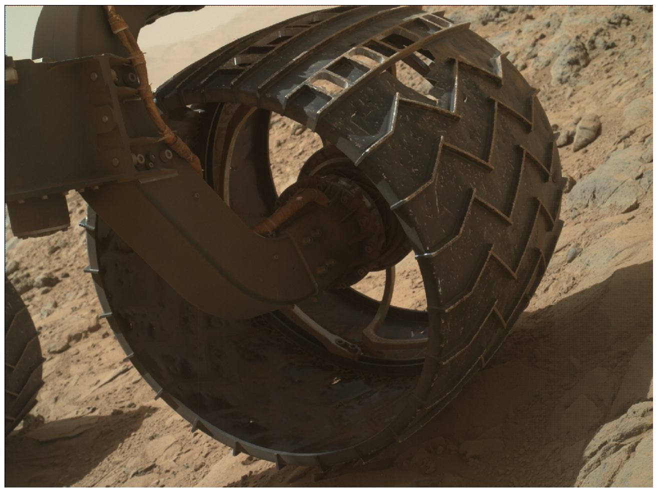


Figure 1. This MAHLI image, taken on Sol 490, depicts puncture damage on the left front wheel.

Prior to the development of this new algorithm, the NASA/JPL Mars rovers moved along a commanded arc by turning each wheel at a constant speed, based on an Ackermann steering model, which assumes the terrain is flat. However, the terrain on Mars is never flat, and ideally, the speed of each wheel should vary based on the local topography of the terrain it is traversing.

To illustrate with a simple example, imagine flat terrain with a single rock ahead of the front left wheel of the rover. As the rover advances, that front wheel needs to climb over the rock, while the other wheels continue to drive on flat ground. In order to climb over the rock, the front wheel must traverse a longer distance than the other five wheels, during the same time period. This means the front wheel should go faster than the other five wheels. Or conversely, since the speed of the wheels is capped, the other five wheels should slow down. Commanding all six wheels to rotate at the same speed results in slip, as the five wheels effectively push the front left wheel forward. This can create damage on the leading wheels if the rock is embedded and sharp, with a tip that can fit between the wheel treads and puncture the wheel skin. As seen in rover images and in ground testing, these punctures initiate the damage for a given skin section and over time, grow to merge with the stress concentration cracking at the tips of the grouser chevrons.

978-1-5386-2014-4/18/$31.00 ©2018 IEEE

Modulating the speed of each wheel to match the terrain topography is a very challenging problem when that topography is unknown. While the Mars rovers can image their surrounding terrain and generate a height map [2], [3], this capability is typically reserved for use in unknown terrain since it a resource-intensive process that significantly reduces overall traverse speed. Moreover, the noise in the height map and accumulated uncertainty in the rover's position and orientation would make it impractical to rely on the terrain mesh to optimize wheel speeds. Instead, we have chosen to develop an approach that only relies on the rover's measured attitude rates and suspension angles (from the rover's gyros and rocker/bogies encoders), and leverages the rigid-body kinematic model of the rover to calculate the optimal wheel speeds as the rover drives.

To implement the algorithm in flight, the team conducted extensive ground testing on both the mobility test vehicle, Scarecrow, and the Vehicle System Testbed (VSTB). Terrains used in this testing were carefully engineered to provide a variety of stressing cases. One of the unexpected behaviors discovered during testing was a "wheelie" observed on the middle wheel of the test vehicle in high-friction terrain. Due to tension in the suspension system, the middle wheel lifted off the terrain and continued to lift. The wheelie was repeatable in a wheel configuration where the front wheel, while descending a rock, began to push against an embedded rock at the same time that the middle wheel crested a rock and the rear wheel was on flat terrain. From this observation, a wheelie suppression behavior was included in the Traction Control algorithm.

The algorithm itself was integrated into the rover's flight software as a hot patch, to be loaded upon each boot. Updates to tactical resource modeling and simulation to account for the increased duration and data volume of the drives were necessary prior to uplink and checkout of the patch on board Curiosity. Similarly, downlink assessment tools had to be updated to view the new data added to the motion history logs. After a three-stage checkout to ensure vehicle safety, the flight performance of the vehicle was closely monitored. Trending results and preliminary flight data confirm the ground performance of the algorithm.

This paper first describes the Traction Control algorithm, then discusses the details of its implementation, including the integration into mission operations, test results, and assessment tools. It concludes with the results of flight performance and plans for long-term trending.

2. ALGORITHM

Our approach consists in using rigid-body kinematics to relate the velocities of each moving part of the rover. The centers of the two front wheels rotate in opposite directions relative to the main body of the rover around the rocker joint. The centers of the middle and rear wheels rotate relative to the rocker body around the left and right bogie joints. Those rotation angles and rates are measured by encoders and enable us to express the wheel velcities as a function of the geometry of the rover, and the measured attitude and suspension rates, and contact angles between the wheels and the terrain.

We first introduce the mathematical framework, including the symbols, coordinate frames, and kinematic formulas. We then describe how we estimate the wheels' contact angles, and conclude with the calculation of the ideal wheel speeds.

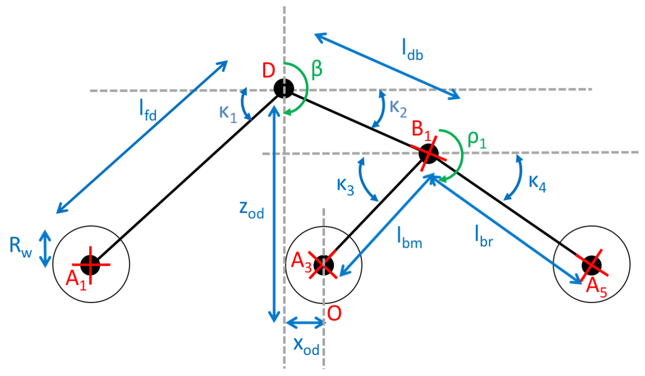


Figure 2. Rover model on flat ground, viewed from the left side.

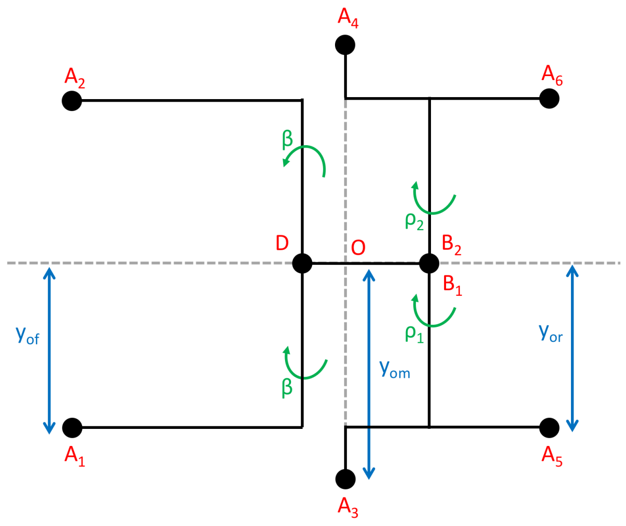


Figure 3. Rover model on flat ground, viewed from above.

Rigid-Body Rover Model

A description of all the symbols used in this section can be found in the *Appendix* section.

We simplify the rover geometry to avoid unnecessary parameters by placing the rocker D and bogies B_1 and B_2 in the x-z plane of the rover's body frame (no lateral offset from the rover's origin O), as illustrated by Figures 2 and 3. The rover origin O is between the middle wheels on the surface, when on flat ground.

Frames

We define the following frames and rotation matrices:

- Body frame bd: follows the aerospace convention, with the x axis along the rover's body length, pointing forward, the y axis pointing to the right of the x axis, and the z axis pointing down.
- Rocker frames rk_1 and rk_2: body frame rotated by the rocker angle ($+\beta$ for rk_1 and $-\beta$ for rk_2). We define the following rotation matrices (from body to rocker frames):

$$^{rk_1}R_{bd} = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \quad (1)$$

$$^{rk_2}R_{bd} = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \quad (2)$$

• Bogie frames bg_1 and bg_2: rocker frames rotated by the bogie angles (ρ_1 for bg_1 and ρ_2 for bg_2). We define the following rotation matrices (from rocker to bogie frames):

$$^{bg_1}R_{rk_1} = \begin{bmatrix} \cos(\rho_1) & 0 & \sin(\rho_1) \\ 0 & 1 & 0 \\ -\sin(\rho_1) & 0 & \cos(\rho_1) \end{bmatrix} \quad (3)$$

$$^{bg_2}R_{rk_2} = \begin{bmatrix} \cos(\rho_2) & 0 & \sin(\rho_2) \\ 0 & 1 & 0 \\ -\sin(\rho_2) & 0 & \cos(\rho_2) \end{bmatrix} \quad (4)$$

• Wheel frames w_i, $i \in [1,6]$: rocker frames (for front wheels) or bogie frames (for middle / rear wheels) rotated by the wheel's steering angle ψ_i. We define the following rotation matrices (from rocker/bogie to wheel frames):

$$^{w_i}R_{rk_i} = \begin{bmatrix} \cos(\psi_i) & -\sin(\psi_i) & 0 \\ \sin(\psi_i) & \cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ for all } i \in \{1,2\} \quad (5)$$

$$^{w_i}R_{bg_1} = \begin{bmatrix} \cos(\psi_i) & -\sin(\psi_i) & 0 \\ \sin(\psi_i) & \cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ for all } i \in \{3,5\} \quad (6)$$

$$^{w_i}R_{bg_2} = \begin{bmatrix} \cos(\psi_i) & -\sin(\psi_i) & 0 \\ \sin(\psi_i) & \cos(\psi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ for all } i \in \{4,6\} \quad (7)$$

• Contact Angle frames η_i, $i \in [1,6]$: wheel frames rotated by the wheel's contact angle η_i. We define the following rotation matrices (from wheel to contact angle frames):

$$^{\eta_i}R_{w_i} = \begin{bmatrix} \cos(\eta_i) & 0 & \sin(\eta_i) \\ 0 & 1 & 0 \\ -\sin(\eta_i) & 0 & \cos(\eta_i) \end{bmatrix} \text{ for all } i \in [1,6] \quad (8)$$

Contact Angle Definition

We define the contact angle η between a wheel and the ground as the angle between the steering actuator axis and the contact point of the wheel and the ground, as illustrated on Figure 4. Note that when the rover is on flat ground, the contact angle for each wheel is zero. If the front wheel climbs over a rock while the other wheels remain on flat ground, the contact angle would become positive as the wheel climbs and negative as the wheel descends.
In reality the wheels of the rover do not make contact with the ground at a single point. This is fine however, as we can still model the motion of the wheel as if there was a single contact point, whose location on the wheel is defined such that the vector from the center of the wheel to the contact point ($\vec{AC}$ in Figure 4) is orthogonal to the velocity vector of the wheel ($\vec{v}$ in 4).

Kinematic Equations

With the parameters of our rover model defined, we can relate the velocities of the wheels to the velocity of any point on the rover using rigid-body kinematics.
The linear velocities of any two points A and B, attached to the same rigid body $\mathcal{R}$, relative to some frame $\mathcal{I}$, and expressed in some arbitrary frame $\mathcal{F}$, are related according to the following key equation:

$$^{\mathcal{F}}\vec{v}_{A/\mathcal{I}} = {}^{\mathcal{F}}\vec{v}_{B/\mathcal{I}} + {}^{\mathcal{F}}\vec{\omega}_{\mathcal{R}/\mathcal{I}} \times {}^{\mathcal{F}}\vec{BA} \quad (9)$$

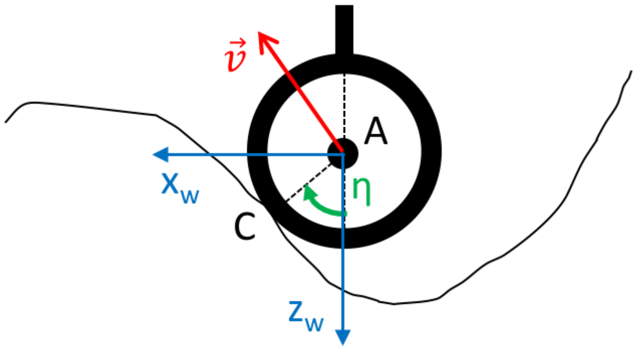


Figure 4. Definition of the contact angle η. $\vec{v}$ is the wheel velocity, $\vec{x_w}$ and $\vec{z_w}$ are the x and z axis of the wheel frame.

Where $^{\mathcal{F}}\vec{\omega}_{\mathcal{R}/\mathcal{I}}$ is the angular velocity vector of the rigid body relative to the frame $\mathcal{I}$, expressed in frame $\mathcal{F}$.

For our purposes, all velocities will be relative to the inertial frame and the angular velocity vector will always be the one of the rover's body relative to the inertial frame, so we'll simplify the notations as follows:

$$^{\mathcal{F}}\vec{v}_A = {}^{\mathcal{F}}\vec{v}_B + {}^{\mathcal{F}}\vec{\omega} \times {}^{\mathcal{F}}\vec{BA} \quad (10)$$

This kinematic relationship can be used to relate the velocities of any two points on the articulated rover body. In order to illustrate our approach, described in details in the following subsections, we show how we can express the velocity of the left front wheel (wheel 1) as a function of the velocity of the rover's origin:

$$^{bd}\vec{v}_O = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

$$^{bd}\vec{v}_D = {}^{bd}\vec{v}_O + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times {}^{bd}\vec{OD} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times {}^{bd}\vec{OD}$$

$$\begin{aligned} ^{bd}\vec{v}_{A_1} &= {}^{bd}\vec{v}_D + \begin{bmatrix} \omega_x \\ \omega_y + \dot{\beta} \\ \omega_z \end{bmatrix} \times {}^{bd}\vec{DA_1} \\ &= \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times {}^{bd}\vec{OD} + \begin{bmatrix} \omega_x \\ \omega_y + \dot{\beta} \\ \omega_z \end{bmatrix} \times {}^{bd}\vec{DA_1} \end{aligned}$$

However, since we want to calculate the ideal (no slip) wheel rates, we need express the wheel velocity in the contact angle frame, since we know that the x component, $^{\eta_1}v^x_{A_1}$, will be proportional to the wheel rate $\dot{\theta}_1$:

$$^{\eta_1}v^x_{A_1} = R_w\, \omega^y_1$$

With ω^y_1 the angular rate of the wheel. However, ω^y_1 is not exactly the same as the angular rate delivered by the drive motor $\dot{\theta}_1$ (what we want to solve for), since that angular rate is relative to the drive actuator, which itself rotates relative to the inertial frame due to the rover's body and suspension rates:

$$\omega_1^y = \dot{\theta}_1 + \zeta_1^y$$

$$\text{with } \vec{\zeta}_1 = {}^{\eta_1}R_{w_1} {}^{w_1}R_{rk_1} {}^{rk_1}R_{bd} \begin{bmatrix} \omega_x \\ \omega_y + \dot{\beta} \\ \omega_z \end{bmatrix}$$

To express the wheel velocity in the contact angle frame, we can use the rotation matrices defined earlier:

$$\begin{aligned} {}^{\eta_1}\vec{v}_{A_1} &= {}^{\eta_1}R_{w_1} {}^{w_1}R_{rk_1} {}^{rk_1}R_{bd} {}^{bd}\vec{v}_{A_1} \\ &= \begin{bmatrix} \cos(\eta_1) & 0 & -\sin(\eta_1) \\ 0 & 1 & 0 \\ \sin(\eta_1) & 0 & \cos(\eta_1) \end{bmatrix} \begin{bmatrix} \cos(\psi_1) & \sin(\psi_1) & 0 \\ -\sin(\psi_1) & \cos(\psi_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &\quad \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \\ &\quad \left(\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times {}^{bd}\vec{OD} + \begin{bmatrix} \omega_x \\ \omega_y + \dot{\beta} \\ \omega_z \end{bmatrix} \times {}^{bd}\vec{DA}_1 \right) \end{aligned}$$

Those equations show that the wheel rate is a function of the attitude and suspension rates, suspension angles, steering angles (all of which are measured), and the wheel's contact angle and rover's linear velocity (not measured).

Algorithm Overview

The functional diagram depicting our technical approach is shown in Figure 5.

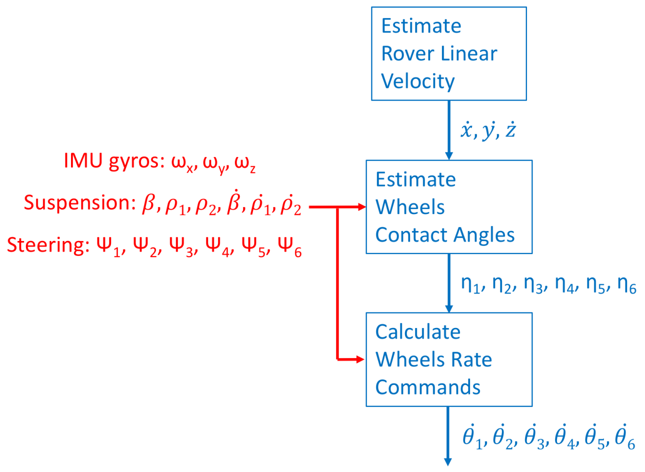

Estimate Rover Linear Velocity
$\dot{x}, \dot{y}, \dot{z}$
IMU gyros: $\omega_x, \omega_y, \omega_z$
Suspension: $\beta, \rho_1, \rho_2, \dot{\beta}, \dot{\rho}_1, \dot{\rho}_2$
Steering: $\Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5, \Psi_6$
Estimate Wheels Contact Angles
$\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6$
Calculate Wheels Rate Commands
$\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5, \dot{\theta}_6$

Figure 5. Functional diagram of the Traction Control algorithm.

Calculation of the Contact Angle Estimates

There are several ways to estimate the contact angles [4]. We chose to trade off some accuracy in favor of robustness by choosing an approach that does not depend on the commanded nor measured wheel velocities. This way, the output of the algorithm, namely the wheel rate commands, do not affect the input of the algorithm at the next time step, thus avoiding any feedback loops that might have caused errors and jeopardized the stability of the algorithm.

A simplifying assumption was made: we approximated the rover's linear velocity to its value on flat ground when calculating the contact angle estimates.

Estimating the Rover's Linear Velocity— The flat ground approximation of the rover's linear velocity in the body frame is:

$${}^{bd}\vec{v}_O = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \dot{x}_0 \\ 0 \\ 0 \end{bmatrix} \tag{11}$$

- For straight driving, all points on the rover have the same speed, which is capped by the maximum wheel rate $\dot{\theta}_{max}$:

$$\dot{x}_0 = \mathsf{dir}\ R_w\ \dot{\theta}_{max} \tag{12}$$

Where dir is the drive direction (+1 for forward, -1 for backward), and R_w is the wheel radius.
- For turns, we are also limited by the maximum wheel rate $\dot{\theta}_{max}$, but $\dot{x}_0$ also depends on the turn radius of the commanded arc, r:

$$\dot{x}_0 = \mathsf{dir}\ R_w\ \dot{\theta}_{max} \frac{|r|}{\max(r_f, r_m, r_r)} \tag{13}$$

Where r_f, r_m, and r_r are the turn radius of the front, middle, and rear wheels, respectively, located to the outside of the turn (away from the center of rotation – those are the wheels with the largest turn radius and therefore going the fastest). Note that r_f, r_m, and r_r are greater than r since the distance from the center of rotation to the rover's origin is always shorter than the distance to the outside wheels, and hence, the velocity of the rover's origin is decreased during turns compared to straight driving. These turn radii of the wheels can be computed as follows:

$$r_f = \sqrt{x_{fm}^2 + (y_{of} + |r|)^2} \tag{14}$$

$$r_m = y_{om} + |r| \tag{15}$$

$$r_r = \sqrt{x_{mr}^2 + (y_{or} + |r|)^2} \tag{16}$$

Estimating the Contact Angles—The contact angle for each wheel is computed by carrying out the following steps:

- Calculate the wheel's linear velocity vector expressed in the wheel's frame.
- Compute the contact angle as the angle between the x and z components of that vector.

We can compute the wheels' linear velocities using the kinematics Equation 10 and our approximation of the linear velocity of the rover's origin (Equation 11). First we compute the linear velocities of the wheels in the body frame:

$$^{bd}v_D = \begin{bmatrix}\dot{x}_0\\0\\0\end{bmatrix} + \begin{bmatrix}\omega_x\\\omega_y\\\omega_z\end{bmatrix} \times {}^{bd}\vec{OD} \tag{17}$$

$$^{bd}v_{A_1} = {}^{bd}v_D - \begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_1D} \tag{18}$$

$$^{bd}v_{A_2} = {}^{bd}v_D - \begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_2D} \tag{19}$$

$$^{bd}v_{B_1} = {}^{bd}v_D - \begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}\\\omega_z\end{bmatrix} \times {}^{bd}\vec{B_1D} \tag{20}$$

$$^{bd}v_{B_2} = {}^{bd}v_D - \begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}\\\omega_z\end{bmatrix} \times {}^{bd}\vec{B_2D} \tag{21}$$

$$^{bd}v_{A_3} = {}^{bd}v_{B_1} - \begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}+\dot{\rho}_1\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_3B_1} \tag{22}$$

$$^{bd}v_{A_4} = {}^{bd}v_{B_2} - \begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}+\dot{\rho}_2\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_4B_2} \tag{23}$$

$$^{bd}v_{A_5} = {}^{bd}v_{B_1} - \begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}+\dot{\rho}_1\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_5B_1} \tag{24}$$

$$^{bd}v_{A_6} = {}^{bd}v_{B_2} - \begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}+\dot{\rho}_2\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_6B_2} \tag{25}$$

With the following parameters based on the rover geometry shown in Figures 2 and 3:

$$^{bd}\vec{OD} = \begin{bmatrix}x_{od}\\0\\z_{od}\end{bmatrix} \tag{26}$$

$$^{bd}\vec{A_1D} = {}^{bd}R_{rk_1}\begin{bmatrix}-l_{fd}\cos(\kappa_1)\\y_{of}\\-l_{fd}\sin(\kappa_1)\end{bmatrix} \tag{27}$$

$$^{bd}\vec{A_2D} = {}^{bd}R_{rk_2}\begin{bmatrix}-l_{fd}\cos(\kappa_1)\\-y_{of}\\-l_{fd}\sin(\kappa_1)\end{bmatrix} \tag{28}$$

$$^{bd}\vec{B_1D} = {}^{bd}R_{rk_1}\begin{bmatrix}l_{db}\cos(\kappa_2)\\0\\-l_{db}\sin(\kappa_2)\end{bmatrix} \tag{29}$$

$$^{bd}\vec{B_2D} = {}^{bd}R_{rk_2}\begin{bmatrix}l_{db}\cos(\kappa_2)\\0\\-l_{db}\sin(\kappa_2)\end{bmatrix} \tag{30}$$

$$^{bd}\vec{A_3B_1} = {}^{bd}R_{rk_1}{}^{rk_1}R_{bg_1}\begin{bmatrix}-l_{bm}\cos(\kappa_3)\\y_{om}\\-l_{bm}\sin(\kappa_3)\end{bmatrix} \tag{31}$$

$$^{bd}\vec{A_4B_2} = {}^{bd}R_{rk_2}{}^{rk_2}R_{bg_2}\begin{bmatrix}-l_{bm}\cos(\kappa_3)\\-y_{om}\\-l_{bm}\sin(\kappa_3)\end{bmatrix} \tag{32}$$

$$^{bd}\vec{A_5B_1} = {}^{bd}R_{rk_1}{}^{rk_1}R_{bg_1}\begin{bmatrix}l_{br}\cos(\kappa_4)\\y_{or}\\-l_{br}\sin(\kappa_4)\end{bmatrix} \tag{33}$$

$$^{bd}\vec{A_6B_2} = {}^{bd}R_{rk_2}{}^{rk_2}R_{bg_2}\begin{bmatrix}l_{br}\cos(\kappa_4)\\-y_{or}\\-l_{br}\sin(\kappa_4)\end{bmatrix} \tag{34}$$

Now we can compute the wheels' linear velocities in the wheels' frames:

$$^{w_1}v_{A_1} = {}^{w_1}R_{rk_1}{}^{rk_1}R_{bd}{}^{bd}v_{A_1} \tag{35}$$
$$^{w_2}v_{A_2} = {}^{w_2}R_{rk_2}{}^{rk_2}R_{bd}{}^{bd}v_{A_2} \tag{36}$$
$$^{w_3}v_{A_3} = {}^{w_3}R_{bg_1}{}^{bg_1}R_{rk_1}{}^{rk_1}R_{bd}{}^{bd}v_{A_3} \tag{37}$$
$$^{w_4}v_{A_4} = {}^{w_4}R_{bg_2}{}^{bg_2}R_{rk_2}{}^{rk_2}R_{bd}{}^{bd}v_{A_4} \tag{38}$$
$$^{w_5}v_{A_5} = {}^{w_5}R_{bg_1}{}^{bg_1}R_{rk_1}{}^{rk_1}R_{bd}{}^{bd}v_{A_5} \tag{39}$$
$$^{w_6}v_{A_6} = {}^{w_6}R_{bg_2}{}^{bg_2}R_{rk_2}{}^{rk_2}R_{bd}{}^{bd}v_{A_6} \tag{40}$$

Finally, we can compute the contact angles as follows:

$$\forall i \in [1,6],\ \eta_i = -\arctan\left(\frac{^{w_i}v^z_{A_i}}{^{w_i}v^x_{A_i}}\right) \tag{41}$$

Calculation of the Wheel Rate Commands

As described in the *Kinematics Equations* subsection, it is possible to relate the wheel angular rates to the linear velocity of the rover origin, contact angles, attitude rates, and suspension angles and rates. Let us derive those equations for all six wheels.

First we can set the y component of the rover's linear velocity to zero since we do not want the rover to move sideways:

$$\dot{y} = 0 \tag{42}$$

This results in:

$$^{bd}\vec{v}_O = \begin{bmatrix}\dot{x}\\0\\\dot{z}\end{bmatrix} \tag{43}$$

Then we compute the wheel linear velocities in the body frame, by applying our kinematics formula (Equation 10) down the chain of linked rigid bodies, starting from the rover's body (at origin O), and moving down to the wheels, passing through the rocker and bogies points:

$$^{bd}v_D = \begin{bmatrix}\dot{x}\\0\\\dot{z}\end{bmatrix} + \begin{bmatrix}\omega_x\\\omega_y\\\omega_z\end{bmatrix} \times {}^{bd}\vec{OD} \quad (44)$$

$$^{bd}v_{A_1} = {}^{bd}v_D - \begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_1D} \quad (45)$$

$$^{bd}v_{A_2} = {}^{bd}v_D - \begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_2D} \quad (46)$$

$$^{bd}v_{B_1} = {}^{bd}v_D - \begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}\\\omega_z\end{bmatrix} \times {}^{bd}\vec{B_1D} \quad (47)$$

$$^{bd}v_{B_2} = {}^{bd}v_D - \begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}\\\omega_z\end{bmatrix} \times {}^{bd}\vec{B_2D} \quad (48)$$

$$^{bd}v_{A_3} = {}^{bd}v_{B_1} - \begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}+\dot{\rho}_1\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_3B_1} \quad (49)$$

$$^{bd}v_{A_4} = {}^{bd}v_{B_2} - \begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}+\dot{\rho}_2\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_4B_2} \quad (50)$$

$$^{bd}v_{A_5} = {}^{bd}v_{B_1} - \begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}+\dot{\rho}_1\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_5B_1} \quad (51)$$

$$^{bd}v_{A_6} = {}^{bd}v_{B_2} - \begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}+\dot{\rho}_2\\\omega_z\end{bmatrix} \times {}^{bd}\vec{A_6B_2} \quad (52)$$

Using the rotation matrices define in the *Frames* subsection, we express the body-frame linear velocities of the wheels in the contact angle frames:

$$^{\eta_1}v_{A_1} = {}^{\eta_1}R_{w_1}{}^{w_1}R_{rk_1}{}^{rk_1}R_{bd}{}^{bd}v_{A_1} \quad (53)$$
$$^{\eta_2}v_{A_2} = {}^{\eta_2}R_{w_2}{}^{w_2}R_{rk_2}{}^{rk_2}R_{bd}{}^{bd}v_{A_2} \quad (54)$$
$$^{\eta_3}v_{A_3} = {}^{\eta_3}R_{w_3}{}^{w_3}R_{bg_1}{}^{bg_1}R_{rk_1}{}^{rk_1}R_{bd}{}^{bd}v_{A_3} \quad (55)$$
$$^{\eta_4}v_{A_4} = {}^{\eta_4}R_{w_4}{}^{w_4}R_{bg_2}{}^{bg_2}R_{rk_2}{}^{rk_2}R_{bd}{}^{bd}v_{A_4} \quad (56)$$
$$^{\eta_5}v_{A_5} = {}^{\eta_5}R_{w_5}{}^{w_5}R_{bg_1}{}^{bg_1}R_{rk_1}{}^{rk_1}R_{bd}{}^{bd}v_{A_5} \quad (57)$$
$$^{\eta_6}v_{A_6} = {}^{\eta_6}R_{w_6}{}^{w_6}R_{bg_2}{}^{bg_2}R_{rk_2}{}^{rk_2}R_{bd}{}^{bd}v_{A_6} \quad (58)$$

We can then relate the wheel rate commands to the x component of the wheels' linear velocities in the contact angle frames:

$$^{\eta_1}v^x_{A_1} = R_w\left(\dot{\theta}_1 + \zeta^y_1\right) \quad (59)$$
$$^{\eta_2}v^x_{A_2} = R_w\left(\dot{\theta}_2 + \zeta^y_2\right) \quad (60)$$
$$^{\eta_3}v^x_{A_3} = R_w\left(\dot{\theta}_3 + \zeta^y_3\right) \quad (61)$$
$$^{\eta_4}v^x_{A_4} = R_w\left(\dot{\theta}_4 + \zeta^y_4\right) \quad (62)$$
$$^{\eta_5}v^x_{A_5} = R_w\left(\dot{\theta}_5 + \zeta^y_5\right) \quad (63)$$
$$^{\eta_6}v^x_{A_6} = R_w\left(\dot{\theta}_6 + \zeta^y_6\right) \quad (64)$$

$$\vec{\zeta}_1 = {}^{\eta_1}R_{w_1}{}^{w_1}R_{rk_1}{}^{rk_1}R_{bd}\begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}\\\omega_z\end{bmatrix} \quad (65)$$

$$\vec{\zeta}_2 = {}^{\eta_2}R_{w_2}{}^{w_2}R_{rk_2}{}^{rk_2}R_{bd}\begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}\\\omega_z\end{bmatrix} \quad (66)$$

$$\vec{\zeta}_3 = {}^{\eta_3}R_{w_3}{}^{w_3}R_{bg_1}{}^{bg_1}R_{rk_1}{}^{rk_1}R_{bd}\begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}+\dot{\rho}_1\\\omega_z\end{bmatrix} \quad (67)$$

$$\vec{\zeta}_4 = {}^{\eta_4}R_{w_4}{}^{w_4}R_{bg_2}{}^{bg_2}R_{rk_2}{}^{rk_2}R_{bd}\begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}+\dot{\rho}_2\\\omega_z\end{bmatrix} \quad (68)$$

$$\vec{\zeta}_5 = {}^{\eta_5}R_{w_5}{}^{w_5}R_{bg_1}{}^{bg_1}R_{rk_1}{}^{rk_1}R_{bd}\begin{bmatrix}\omega_x\\\omega_y+\dot{\beta}+\dot{\rho}_1\\\omega_z\end{bmatrix} \quad (69)$$

$$\vec{\zeta}_6 = {}^{\eta_6}R_{w_6}{}^{w_6}R_{bg_2}{}^{bg_2}R_{rk_2}{}^{rk_2}R_{bd}\begin{bmatrix}\omega_x\\\omega_y-\dot{\beta}+\dot{\rho}_2\\\omega_z\end{bmatrix} \quad (70)$$

Which, after leveraging Equations 1 - 8 provides:

$$\begin{aligned}
\zeta^y_1 &= (\omega_y + \dot{\beta})\cos(\psi_1) \\
&\quad - (\omega_x\cos(\beta) - \omega_z\sin(\beta))\sin(\psi_1) && (71)\\
\zeta^y_2 &= (\omega_y - \dot{\beta})\cos(\psi_2) \\
&\quad - (\omega_x\cos(\beta) + \omega_z\sin(\beta))\sin(\psi_2) && (72)\\
\zeta^y_3 &= \omega_x + \dot{\beta} + \dot{\rho}_1 && (73)\\
\zeta^y_4 &= \omega_x - \dot{\beta} + \dot{\rho}_2 && (74)\\
\zeta^y_5 &= (\omega_x + \dot{\beta} + \dot{\rho}_1)\cos(\psi_5) \\
&\quad - (\omega_x\cos(\beta+\rho_1) - \omega_z\sin(\beta+\rho_1))\sin(\psi_5) && (75)\\
\zeta^y_6 &= (\omega_x - \dot{\beta} + \dot{\rho}_2)\cos(\psi_6) \\
&\quad - (\omega_x\cos(\rho_2-\beta) - \omega_z\sin(\rho_2-\beta))\sin(\psi_6) && (76)
\end{aligned}$$

Substituting those variables into the right sides of Equations 59 - 64 and expanding the left side using the prior equations enables us to obtain the desired wheel rate equations:

$$\begin{aligned}
\dot{\theta}_1 &= ((\cos(\beta)\sin(\eta_1) + \cos(\eta_1)\cos(\psi_1)\sin(\beta)) \\
&\quad (\omega_y x_{od} - \dot{z} + \omega_x y_{of} + l_{fd}\cos(\beta-\kappa_1)(\dot{\beta}+\omega_y)) \\
&\quad - (\sin(\beta)\sin(\eta_1) - \cos(\beta)\cos(\eta_1)\cos(\psi_1))(\dot{x}+ \\
&\quad \omega_y z_{od} + y_{of}\omega_z - l_{fd}\sin(\beta-\kappa_1)(\dot{\beta}+\omega_y)) + R_w \\
&\quad (\cos(\psi_1)(\dot{\beta}+\omega_y) - \omega_x\cos(\beta)\sin(\psi_1) + \omega_z\sin(\beta) \\
&\quad \sin(\psi_1)) + \cos(\eta_1)\sin(\psi_1)(-\omega_x z_{od} + x_{od}\omega_z \\
&\quad + l_{fd}\omega_z\cos(\beta-\kappa_1) + l_{fd}\omega_x\sin(\beta-\kappa_1)))/R_w && (77)\\
\dot{\theta}_2 &= ((\sin(\beta)\sin(\eta_2) + \cos(\beta)\cos(\eta_2)\cos(\psi_2)) \\
&\quad (\dot{x} + \omega_y z_{od} - y_{of}\omega_z - l_{fd}\sin(\beta+\kappa_1)(\dot{\beta}-\omega_y)) \\
&\quad - R_w(\cos(\psi_2)(\dot{\beta}-\omega_y) + \omega_x\cos(\beta)\sin(\psi_2) \\
&\quad + \omega_z\sin(\beta)\sin(\psi_2)) - (\cos(\beta)\sin(\eta_2) - \cos(\eta_2) \\
&\quad \cos(\psi_2)\sin(\beta))(\dot{z} - \omega_y x_{od} + \omega_x y_{of} + l_{fd}\cos(\beta+\kappa_1) \\
&\quad (\dot{\beta}-\omega_y)) + \cos(\eta_2)\sin(\psi_2)(x_{od}\omega_z - \omega_x z_{od} \\
&\quad + l_{fd}\omega_z\cos(\beta+\kappa_1) - l_{fd}\omega_x\sin(\beta+\kappa_1)))/R_w && (78)
\end{aligned}$$

$$\begin{aligned}\dot{\theta}_3 &= (R_w(\dot{\beta} + \omega_y + \dot{\rho}_1) + \dot{x}\cos(\beta + \eta_3 + \rho_1)\\ &- \dot{z}\sin(\beta + \eta_3 + \rho_1) + \dot{\beta} l_{db}\sin(\kappa_2 - \eta_3 - \rho_1)\\ &+ l_{db}\omega_y \sin(\kappa_2 - \eta_3 - \rho_1) + \omega_y z_{od}\cos(\beta + \eta_3 + \rho_1)\\ &+ y_{om}\omega_z\cos(\beta + \eta_3 + \rho_1) + \omega_y x_{od}\sin(\beta + \eta_3 + \rho_1)\\ &+ \omega_x y_{om}\sin(\beta + \eta_3 + \rho_1) + \dot{\beta} l_{bm}\sin(\kappa_3 + \eta_3)\\ &+ l_{bm}\omega_y\sin(\kappa_3 + \eta_3) + l_{bm}\dot{\rho}_1\sin(\kappa_3 + \eta_3))/R_w \quad (79)\end{aligned}$$

$$\begin{aligned}\dot{\theta}_4 &= (\dot{x}\cos(\beta - \eta_4 - \rho_2) + \dot{z}\sin(\beta - \eta_4 - \rho_2)\\ &+ R_w(\omega_y - \dot{\beta} + \dot{\rho}_2) - \dot{\beta} l_{db}\sin(\kappa_2 - \eta_4 - \rho_2)\\ &+ \omega_y z_{od}\cos(\beta - \eta_4 - \rho_2) - y_{om}\omega_z\cos(\beta - \eta_4 - \rho_2)\\ &+ l_{db}\omega_y\sin(\kappa_2 - \eta_4 - \rho_2) - \omega_y x_{od}\sin(\beta - \eta_4 - \rho_2)\\ &+ \omega_x y_{om}\sin(\beta - \eta_4 - \rho_2) - \dot{\beta} l_{bm}\sin(\kappa_3 + \eta_4)\\ &+ l_{bm}\omega_y\sin(\kappa_3 + \eta_4) + l_{bm}\dot{\rho}_2\sin(\kappa_3 + \eta_4))/R_w \quad (80)\end{aligned}$$

$$\begin{aligned}\dot{\theta}_5 &= -((\cos(\beta)(\cos(\rho_1)\sin(\eta_5) + \cos(\eta_5)\cos(\psi_5)\sin(\rho_1))\\ &- \sin(\beta)(\sin(\eta_5)\sin(\rho_1) - \cos(\eta_5)\cos(\psi_5)\cos(\rho_1)))\\ &(\dot{z} - \omega_y x_{od} - \omega_x y_o r + \dot{\beta} l_{br}\cos(\beta + \kappa_4 + \rho_1)\\ &+ l_{br}\omega_y\cos(\beta + \kappa_4 + \rho_1) + l_{br}\dot{\rho}_1\cos(\beta + \kappa_4 + \rho_1)\\ &+ \dot{\beta} l_{db}\cos(\beta + \kappa_2) + l_{db}\omega_y\cos(\beta + \kappa_2))\\ &- R_w(\cos(\psi_5)(\dot{\beta} + \omega_y + \dot{\rho}_1) - \omega_x\cos(\beta + \rho_1)\sin(\psi_5)\\ &+ \omega_z\sin(\beta + \rho_1)\sin(\psi_5)) + (\cos(\beta)(\sin(\eta_5)\sin(\rho_1)\\ &- \cos(\eta_5)\cos(\psi_5)\cos(\rho_1)) + \sin(\beta)(\cos(\rho_1)\sin(\eta_5)\\ &+ \cos(\eta_5)\cos(\psi_5)\sin(\rho_1)))(\dot{x} + \omega_y z_{od} + y_{or}\omega_z\\ &+ \dot{\beta} l_{br}\sin(\beta + \kappa_4 + \rho_1) + l_{br}\omega_y\sin(\beta + \kappa_4 + \rho_1)\\ &+ l_{br}\dot{\rho}_1\sin(\beta + \kappa_4 + \rho_1) + \dot{\beta} l_{db}\sin(\beta + \kappa_2)\\ &+ l_{db}\omega_y\sin(\beta + \kappa_2)) + \cos(\eta_5)\sin(\psi_5)(\omega_x z_{od} - x_{od}\omega_z\\ &+ l_{br}\omega_z\cos(\beta + \kappa_4 + \rho_1) + l_{br}\omega_x\sin(\beta + \kappa_4 + \rho_1)\\ &+ l_{db}\omega_z\cos(\beta + \kappa_2) + l_{db}\omega_x\sin(\beta + \kappa_2)))/R_w \quad (81)\end{aligned}$$

$$\begin{aligned}\dot{\theta}_6 &= -((\cos(\beta)(\cos(\rho_2)\sin(\eta_6) + \cos(\eta_6)\cos(\psi_6)\sin(\rho_2))\\ &+ \sin(\beta)(\sin(\eta_6)\sin(\rho_2) - \cos(\eta_6)\cos(\psi_6)\cos(\rho_2)))\\ &(\dot{z} - \omega_y x_{od} + \omega_x y_o r + l_{br}\dot{\rho}_2\cos(\beta - \kappa_4 - \rho_2)\\ &- \dot{\beta} l_{db}\cos(\beta - \kappa_2) + l_{db}\omega_y\cos(\beta - \kappa_2)\\ &- \dot{\beta} l_{br}\cos(\beta - \kappa_4 - \rho_2) + l_{br}\omega_y\cos(\beta - \kappa_4 - \rho_2))\\ &+ R_w(\omega_x\sin(\psi_6)\cos(\beta - \rho_2) - \cos(\psi_6)(\omega_y - \dot{\beta} + \dot{\rho}_2)\\ &+ \omega_z\sin(\beta - \rho_2)\sin(\psi_6)) + (\cos(\beta)(\sin(\eta_6)\sin(\rho_2)\\ &- \cos(\eta_6)\cos(\psi_6)\cos(\rho_2)) - \sin(\beta)(\cos(\rho_2)\sin(\eta_6)\\ &+ \cos(\eta_6)\cos(\psi_6)\sin(\rho_2)))(\dot{x} + \omega_y z_{od} - y_{or}\omega_z\\ &+ \dot{\beta} l_{br}\sin(\beta - \kappa_4 - \rho_2) - l_{br}\omega_y\sin(\beta - \kappa_4 - \rho_2)\\ &- l_{br}\dot{\rho}_2\sin(\beta - \kappa_4 - \rho_2) + \dot{\beta} l_{db}\sin(\beta - \kappa_2)\\ &- l_{db}\omega_y\sin(\beta - \kappa_2)) - \cos(\eta_6)\sin(\psi_6)(x_{od}\omega_z - \omega_x z_{od}\\ &- l_{br}\omega_z\cos(\beta - \kappa_4 - \rho_2) + l_{br}\omega_x\sin(\beta - \kappa_4 - \rho_2)\\ &- l_{db}\omega_z\cos(\beta - \kappa_2) + l_{db}\omega_x\sin(\beta - \kappa_2)))/R_w \quad (82)\end{aligned}$$

Hence we can compute the wheel rate commands to achieve a desired rover motion (body linear velocity and heading rate). However, the mobility system of the Mars rover is not designed to achieve a pre-determined rover velocity, nor heading rate. Instead, it drives the wheels as fast as possible, i.e. setting at least one wheel to its maximum rate $\dot{\theta}_{max}$, based on the arc being driven which defines the proportion of longitudinal motion and heading change (i.e. the ratio of $\dot{x}$ and ω_z).

Our approach is to turn the wheel rate equations around to express the rover's linear velocity (and heading rate for turns) as a function of the wheel rate commands, then set all wheel rate commands to the max value $\dot{\theta}_{max}$, and determine the minimum absolute value of the computed rover velocity (or heading rate for turns) over all wheels. This technique allows us to determine which wheel gets to its max angular rate first and deduce the associated maximum rover velocity and heading rate.

To do this we first need to relate $\dot{z}$, ω_z, and $\dot{x}$ since a single (wheel rate) equation can only solve for a single unknown.

- For straight driving, our desired heading rate is zero:

$$\omega_z = 0 \quad (83)$$

Thus we only need to express $\dot{z}$ as a function of $\dot{x}$.
- For turns, we chose to solve for the heading rate ω_z rather than $\dot{x}$, since the latter is zero for the special case of turns in place (where the turn radius is zero). We can use the known turn radius r to relate $\dot{x}$ to ω_z:

$$\dot{x} = r\ \omega_z \quad (84)$$

Then, if we can express $\dot{z}$ as a function of $\dot{x}$, like in the straight driving case, we can also express $\dot{z}$ as a function of ω_z (using Equation 84).

To express $\dot{z}$ as a function of $\dot{x}$, we compute the linear velocity of each wheel in its contact angle frame, and leverage the fact that the z component, which depends on both $\dot{x}$ and $\dot{z}$, is zero (since by definition the contact angle frame is rotated so that the wheel velocity vector in the x-z plane is along the x axis only):

$$\forall i \in [1,6],\ {}^{\eta_i}v^z_{A_i} = 0 \quad (85)$$

Expanding Equations 53 - 58 and applying Equation 85 above to their z components, we get:

$$\forall i \in [1,6],\ a_{i1}\dot{x} + a_{i2}\dot{z} + a_{i3}\omega_z = b_i \quad (86)$$

With:

$$\begin{aligned}a_{11} &= \cos(\eta_1)\sin(\beta) + \cos(\beta)\cos(\psi_1)\sin(\eta_1) \quad (87)\\ a_{12} &= \cos(\beta)\cos(\eta_1) - \cos(\psi_1)\sin(\beta)\sin(\eta_1) \quad (88)\\ a_{13} &= y_{of}(\cos(\eta_1)\sin(\beta) + \cos(\beta)\cos(\psi_1)\sin(\eta_1))\\ &+ \sin(\eta_1)\sin(\psi_1)(x_{od} + l_{fd}\cos(\beta - \kappa_1)) \quad (89)\end{aligned}$$

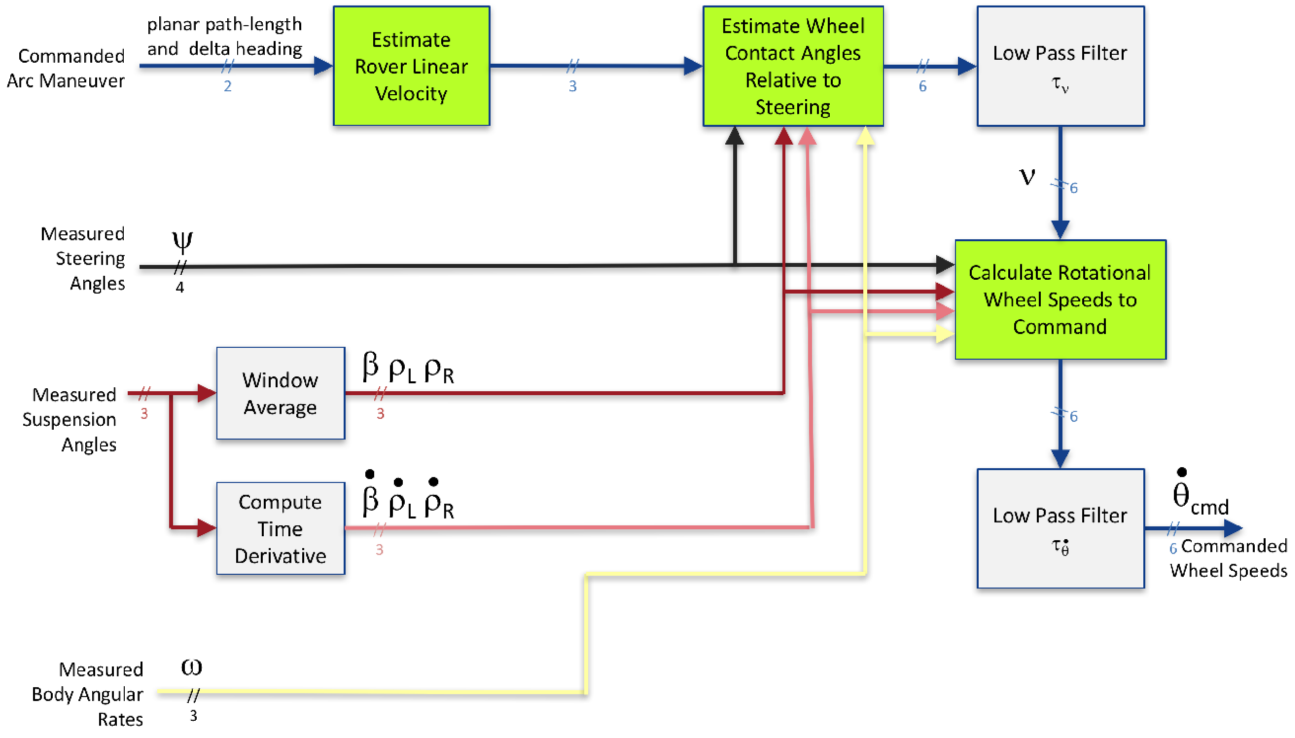

Commanded Arc Maneuver
planar path-length and delta heading
2
Estimate Rover Linear Velocity
3
Estimate Wheel Contact Angles Relative to Steering
6
Low Pass Filter
τ_ν
ν
6
Measured Steering Angles
ψ
4
Calculate Rotational Wheel Speeds to Command
6
Measured Suspension Angles
3
Window Average
$\beta\ \rho_L\ \rho_R$
3
Compute Time Derivative
$\dot\beta\ \dot\rho_L\ \dot\rho_R$
3
Low Pass Filter
$\tau_{\dot\theta}$
$\dot\theta_{cmd}$
6
Commanded Wheel Speeds
Measured Body Angular Rates
ω
3

Figure 6. Data flow for the Traction Control algorithm implementation.

$$
\begin{aligned}
b_1 = {} & (\cos(\beta)\cos(\eta_1) - \cos(\psi_1)\sin(\beta)\sin(\eta_1)) \\
& (\omega_y x_{od} + \omega_x y_{of} + l_{fd}\cos(\beta - \kappa_1) \\
& (\dot\beta + \omega_y)) - (\cos(\eta_1)\sin(\beta) + \cos(\beta)\cos(\psi_1)\sin(\eta_1)) \\
& (\omega_y z_{od} - l_{fd}\sin(\beta - \kappa_1)(\dot\beta + \omega_y)) \\
& + \sin(\eta_1)\sin(\psi_1)(\omega_x z_{od} - l_{fd}\omega_x \sin(\beta - \kappa_1)) && (90)
\end{aligned}
$$

$$a_{21} = \cos(\beta)\cos(\psi_2)\sin(\eta_2) - \cos(\eta_2)\sin(\beta) \tag{91}$$

$$a_{22} = \cos(\beta)\cos(\eta_2) + \cos(\psi_2)\sin(\beta)\sin(\eta_2) \tag{92}$$

$$
\begin{aligned}
a_{23} = {} & y_{of}(\cos(\eta_2)\sin(\beta) - \cos(\beta)\cos(\psi_2)\sin(\eta_2)) \\
& + \sin(\eta_2)\sin(\psi_2)(x_{od} + l_{fd}\cos(\beta + \kappa_1)) && (93)
\end{aligned}
$$

$$
\begin{aligned}
b_2 = {} & (\cos(\eta_2)\sin(\beta) - \cos(\beta)\cos(\psi_2)\sin(\eta_2)) \\
& (\omega_y z_{od} - l_{fd}\sin(\beta + \kappa_1)(\dot\beta - \omega_y)) \\
& - (\cos(\beta)\cos(\eta_2) + \cos(\psi_2)\sin(\beta)\sin(\eta_2)) \\
& (\omega_x y_{of} - \omega_y x_{od} + l_{fd}\cos(\beta + \kappa_1)(\dot\beta - \omega_y)) \\
& + \sin(\eta_2)\sin(\psi_2)(\omega_x z_{od} + l_{fd}\omega_x \sin(\beta + \kappa_1)) && (94)
\end{aligned}
$$

$$a_{31} = \sin(\beta + \eta_3 + \rho_1) \tag{95}$$

$$a_{32} = \cos(\beta + \eta_3 + \rho_1) \tag{96}$$

$$a_{33} = y_{om}\sin(\beta + \eta_3 + \rho_1) \tag{97}$$

$$
\begin{aligned}
b_3 = {} & \omega_y x_{od}\cos(\beta + \eta_3 + \rho_1) + \omega_x y_{om}\cos(\beta + \eta_3 + \rho_1) \\
& - \omega_y z_{od}\sin(\beta + \eta_3 + \rho_1) - \dot\beta l_{db}\cos(\eta_3 - \kappa_2 + \rho_1) \\
& - l_{db}\omega_y\cos(\eta_3 - \kappa_2 + \rho_1) + \dot\beta l_{bm}\cos(\eta_3 + \kappa_3) \\
& + l_{bm}\omega_y\cos(\eta_3 + \kappa_3) + l_{bm}\dot\rho_1\cos(\eta_3 + \kappa_3) && (98)
\end{aligned}
$$

$$a_{41} = \sin(\eta_4 - \beta + \rho_2) \tag{99}$$

$$a_{42} = \cos(\eta_4 - \beta + \rho_2) \tag{100}$$

$$a_{43} = -y_{om}\sin(\eta_4 - \beta + \rho_2) \tag{101}$$

$$
\begin{aligned}
b_4 = {} & \dot\beta l_{db}\cos(\eta_4 - \kappa_2 + \rho_2) - l_{db}\omega_y\cos(\eta_4 - \kappa_2 + \rho_2) \\
& + \omega_y x_{od}\cos(\eta_4 - \beta + \rho_2) - \omega_x y_{om}\cos(\eta_4 - \beta + \rho_2) \\
& - \omega_y z_{od}\sin(\eta_4 - \beta + \rho_2) - \dot\beta l_{bm}\cos(\eta_4 + \kappa_3) \\
& + l_{bm}\omega_y\cos(\eta_4 + \kappa_3) + l_{bm}\dot\rho_2\cos(\eta_4 + \kappa_3) && (102)
\end{aligned}
$$

$$
\begin{aligned}
a_{51} = {} & \cos(\beta)(\cos(\eta_5)\sin(\rho_1) + \cos(\psi_5)\cos(\rho_1)\sin(\eta_5)) \\
& + \sin(\beta)(\cos(\eta_5)\cos(\rho_1) - \cos(\psi_5)\sin(\eta_5)\sin(\rho_1)) && (103)
\end{aligned}
$$

$$
\begin{aligned}
a_{52} = {} & \cos(\beta)(\cos(\eta_5)\cos(\rho_1) - \cos(\psi_5)\sin(\eta_5)\sin(\rho_1)) \\
& - \sin(\beta)(\cos(\eta_5)\sin(\rho_1) + \cos(\psi_5)\cos(\rho_1)\sin(\eta_5)) && (104)
\end{aligned}
$$

$$
\begin{aligned}
a_{53} = {} & y_{or}(\cos(\beta)(\cos(\eta_5)\sin(\rho_1) + \cos(\psi_5)\cos(\rho_1)\sin(\eta_5)) \\
& + \sin(\beta)(\cos(\eta_5)\cos(\rho_1) - \cos(\psi_5)\sin(\eta_5)\sin(\rho_1))) \\
& - \sin(\eta_5)\sin(\psi_5)(l_{db}\cos(\beta + \kappa_2) - x_{od} \\
& + l_{br}\cos(\beta + \kappa_4 + \rho_1)) && (105)
\end{aligned}
$$

$$
\begin{aligned}
b_5 = {} & \omega_x\sin(\eta_5)\sin(\psi_5)(z_{od} + l_{db}\sin(\beta + \kappa_2) \\
& + l_{br}\sin(\beta + \kappa_4 + \rho_1)) - (\cos(\beta)(\cos(\eta_5)\sin(\rho_1) \\
& + \cos(\psi_5)\cos(\rho_1)\sin(\eta_5)) + \sin(\beta)(\cos(\eta_5)\cos(\rho_1) \\
& - \cos(\psi_5)\sin(\eta_5)\sin(\rho_1)))(\omega_y z_{od} + \dot\beta l_{br}\sin(\beta + \kappa_4 \\
& + \rho_1) + l_{br}\omega_y\sin(\beta + \kappa_4 + \rho_1) + l_{br}\dot\rho_1\sin(\beta + \kappa_4 \\
& + \rho_1) + \dot\beta l_{db}\sin(\beta + \kappa_2) + l_{db}\omega_y\sin(\beta + \kappa_2)) \\
& - (\cos(\beta)(\cos(\eta_5)\cos(\rho_1) - \cos(\psi_5)\sin(\eta_5)\sin(\rho_1)) \\
& - \sin(\beta)(\cos(\eta_5)\sin(\rho_1) + \cos(\psi_5)\cos(\rho_1)\sin(\eta_5))) \\
& (\dot\beta l_{br}\cos(\beta + \kappa_4 + \rho_1) - \omega_x y_{or} - \omega_y x_{od} \\
& + l_{br}\omega_y\cos(\beta + \kappa_4 + \rho_1) + l_{br}\dot\rho_1\cos(\beta + \kappa_4 + \rho_1) \\
& + \dot\beta l_{db}\cos(\beta + \kappa_2) + l_{db}\omega_y\cos(\beta + \kappa_2)) && (106)
\end{aligned}
$$

$$a_{61} = \cos(\beta)(\cos(\eta_6)\sin(\rho_2) + \cos(\psi_6)\cos(\rho_2)\sin(\eta_6)) - \sin(\beta)(\cos(\eta_6)\cos(\rho_2) - \cos(\psi_6)\sin(\eta_6)\sin(\rho_2)) \tag{107}$$

$$a_{62} = \cos(\beta)(\cos(\eta_6)\cos(\rho_2) - \cos(\psi_6)\sin(\eta_6)\sin(\rho_2)) + \sin(\beta)(\cos(\eta_6)\sin(\rho_2) + \cos(\psi_6)\cos(\rho_2)\sin(\eta_6)) \tag{108}$$

$$\begin{aligned} a_{63} = -y_{or}(&\cos(\beta)(\cos(\eta_6)\sin(\rho_2) \\ &+ \cos(\psi_6)\cos(\rho_2)\sin(\eta_6)) - \sin(\beta)(\cos(\eta_6)\cos(\rho_2) \\ &- \cos(\psi_6)\sin(\eta_6)\sin(\rho_2))) - \sin(\eta_6)\sin(\psi_6) \\ &(l_{br}\cos(\kappa_4 - \beta + \rho_2) - x_{od} + l_{db}\cos(\beta - \kappa_2)) \end{aligned} \tag{109}$$

$$\begin{aligned} b_6 = \sin(\eta_6)&\sin(\psi_6)(\omega_x z_{od} - l_{db}\omega_x \sin(\beta - \kappa_2) \\ &+ l_{br}\omega_x \sin(\kappa_4 - \beta + \rho_2)) - (\cos(\beta)(\cos(\eta_6)\sin(\rho_2) \\ &+ \cos(\psi_6)\cos(\rho_2)\sin(\eta_6)) - \sin(\beta)(\cos(\eta_6)\cos(\rho_2) \\ &- \cos(\psi_6)\sin(\eta_6)\sin(\rho_2)))(\omega_y z_{od} + \dot{\beta} l_{db}\sin(\beta - \kappa_2) \\ &- l_{db}\omega_y \sin(\beta - \kappa_2) - \dot{\beta} l_{br}\sin(\kappa_4 - \beta + \rho_2) \\ &+ l_{br}\omega_y \sin(\kappa_4 - \beta + \rho_2) + l_{br}\dot{\rho}_2 \sin(\kappa_4 - \beta + \rho_2)) \\ &- (\cos(\beta)(\cos(\eta_6)\cos(\rho_2) - \cos(\psi_6)\sin(\eta_6)\sin(\rho_2)) \\ &+ \sin(\beta)(\cos(\eta_6)\sin(\rho_2) + \cos(\psi_6)\cos(\rho_2)\sin(\eta_6))) \\ &(\omega_x y_{or} - \omega_y x_{od} - \dot{\beta} l_{db}\cos(\beta - \kappa_2) \\ &+ l_{db}\omega_y \cos(\beta - \kappa_2) - \dot{\beta} l_{br}\cos(\kappa_4 - \beta + \rho_2) \\ &+ l_{br}\omega_y \cos(\kappa_4 - \beta + \rho_2) + l_{br}\dot{\rho}_2 \cos(\kappa_4 - \beta + \rho_2)) \end{aligned} \tag{110}$$

Based on the above we define:

- For straight driving ($\omega_z = 0$), for wheel i:

$$\dot{z} = c_i - d_i \dot{x} \tag{111}$$

With:

$$c_i = \frac{b_i}{a_{i2}} \tag{112}$$

$$d_i = \frac{a_{i1}}{a_{i2}} \tag{113}$$

- For turns of turn radius r ($\dot{x} = r\ \omega_z$), for wheel i:

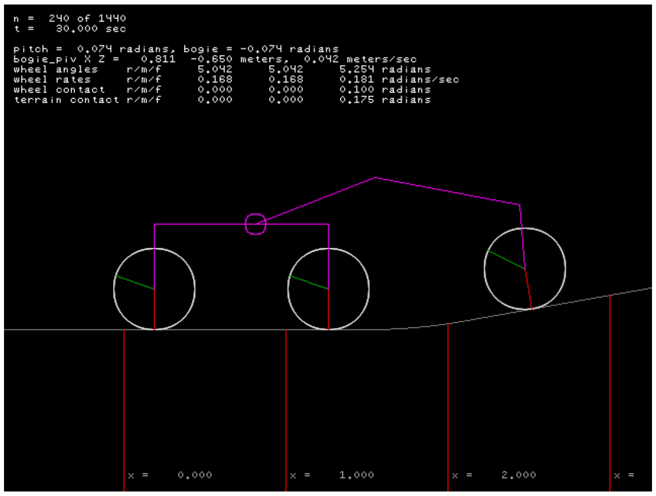
$$\dot{z} = c_i - e_i \omega_z \tag{114}$$

With:

$$e_i = \frac{a_{i3} + r\ a_{i1}}{a_{i2}} \tag{115}$$

Note that we really only need one of the equations relating $\dot{z}$ to $\dot{x}$ (from Equation 111) and $\dot{z}$ to ω_z (from Equation 114). However, using the equation associated with the same wheel for which we are trying to compute $\dot{x}$ and ω_z does simplify the formulas. In particular it removes some denominators, and hence the risk of dividing by zero, and makes the expressions of the remaining denominators easier to interpret (they become zero if and only if the associated wheel is perpendicular to the direction of motion).

We then use Equations 83 - 115 to substitute ω_z and $\dot{z}$ in Equations 77 - 82 and compute the $\dot{x}$ for straight driving, and ω_z for turns, associated with each wheel turning at its max rate $\mathsf{dir}\ \dot{\theta}_{max}$ (where dir represents the direction of rotation of the wheel).



```
n =  240 of 1440
t =   30.000 sec

pitch =  0.074 radians, bogie = -0.074 radians
bogie_piv X Z =   0.811  -0.650 meters,  0.042 meters/sec
wheel angles    r/m/f     5.042     5.042     5.254 radians
wheel rates     r/m/f     0.168     0.168     0.181 radians/sec
wheel contact   r/m/f     0.000     0.000     0.100 radians
terrain contact r/m/f     0.000     0.000     0.175 radians
```

Figure 7. 2D Simulator used to validate our algorithm.

The algorithm to calculate the commanded wheel rates can be summarized as follows:

(1) Compute the desired $\dot{x}$, $\dot{z}$, and ω_z that brings at least one wheel to its maximum rate.

- For straight driving:

$$\begin{cases} i_{min} = \arg\min_{i \in [1,6]} |\dot{x}^i| \\ \dot{x} = \dot{x}^{i_{min}} \\ \dot{z} = c_{i_{min}} - d_{i_{min}}\dot{x}, \text{ using Equations 87 - 113} \\ \omega_z = 0 \end{cases}$$

- For turns:

$$\begin{cases} i_{min} = \arg\min_{i \in [1,6]} |\omega_z^i| \\ \omega_z = \omega_z^{i_{min}} \\ \dot{x} = r\ \omega_z \\ \dot{z} = c_{i_{min}} - e_{i_{min}}\omega_z \text{ , using Equations 87 - 115} \end{cases}$$

(2) Calculate the wheel rate commands using Equations 77 - 82.

Figure 6 shows the functional block diagram associated with the implementation of the entire Traction Control algorithm.

Simulation Testing

Both a 2D and a 3D motion simulator were developed to verify the output of the algorithm (see Figure 7). For the 2D case, the ideal "no-slip" wheel rates were calculated from the equation of the terrain curve. For the 3D case, we used a straight line distance approximation by settling the rover on the terrain at very short spatial intervals. In both cases we verified that the simulated, ideal wheel rates match our algorithm's output when we remove the approximation on the linear velocity of the rover origin (see the *Calculation of the Contact Angle Estimates* subsection) and instead use the correct rover velocity, which can be computed in simulation.

Figure 8 shows a comparison of the ideal wheel rates in simulation (left), the wheel rates generated by our algorithm in simulation (middle), and the wheel rates measured on our rover testbed when commanded by our algorithm (right), for a straight arc test case where the right front wheel of the rover drives over a rock while the others remain on flat ground.

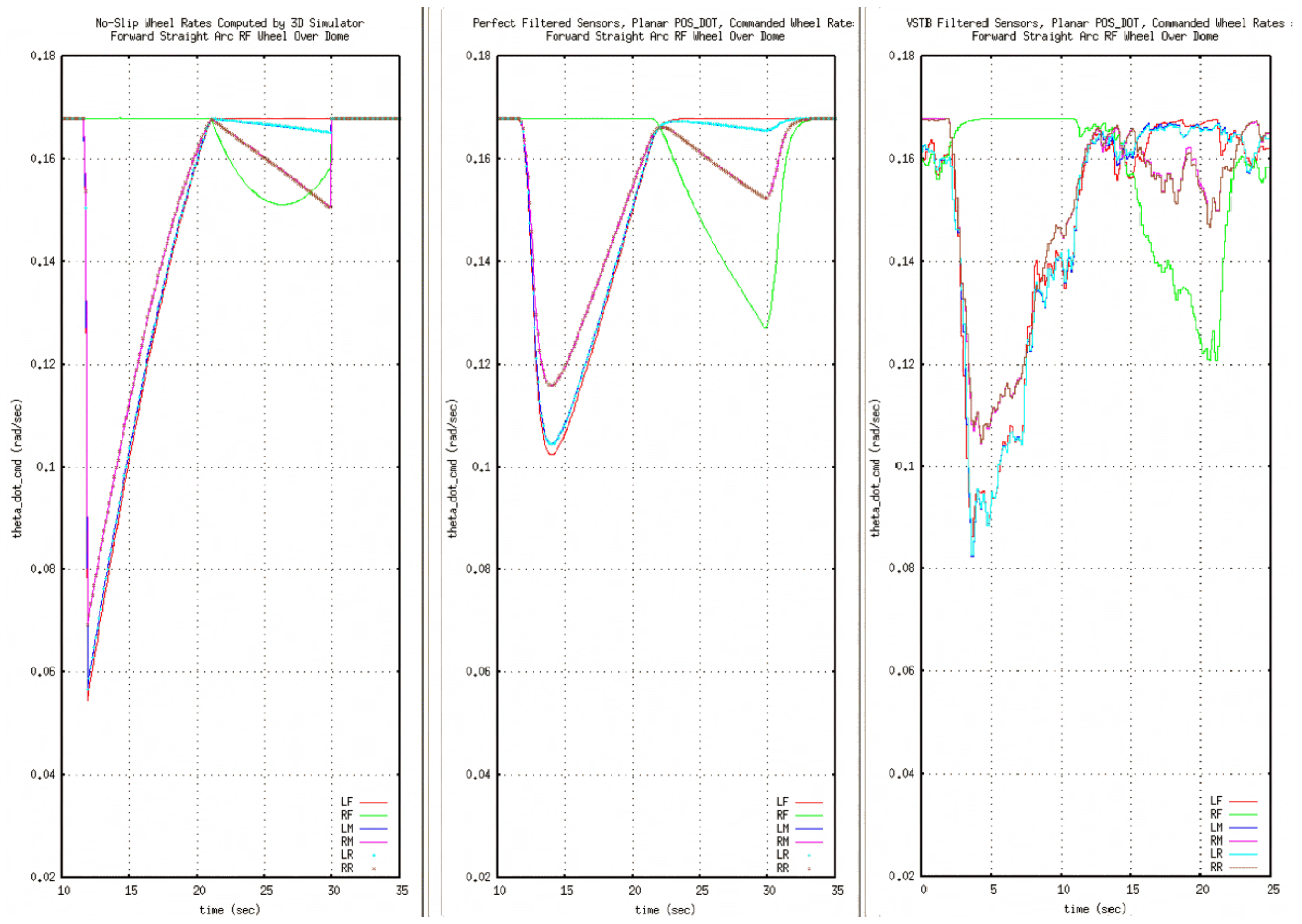


Figure 8. Comparison of the wheel rates obtained in simulation with perfect knowledge (left), output by our algorithm in simulation (middle), and measured in the rover testbed (VSTB) when using our algorithm (right).

3. Performance in Test

In this section the results of tests performed during both development and validation and verification (V&V) are discussed. The example results presented emphasize the reduction of loads and drive actuator torques and are further broken down into three subsections: i) development tests performed on benign, single obstacle terrain, ii) tests performed over *complex* terrain, and iii) yaw reduction. Here, the term *complex* is used to describe a series of decimeter-scale obstacles, which when traversed, yield complex vehicle-terrain interaction in which measurements made by the IMU and suspension resolvers are a function of multiple ground interactions. It should be noted that the majority of loads data gathered during this task were collected to enable the drawing of qualitative conclusions regarding the efficacy of Traction Control. That is, due to the complexity and range of all possible terrains Curiosity may encounter over the course of its remaining life, an attempt to quantify the precise load reduction on one, or a small subset of, these terrains was not performed. Rather, the test philosophy adopted here was to perform tests over a wide array of terrains under various driving conditions such that the expected average load reduction could be approximated.

Benign Single Obstacle Terrain Testing

The Scarecrow rover, shown in Figure 9, was developed in support of MSL mobility system performance evaluation. The Scarecrow mobility system (wheels, rocker, bogies, and differential) is kinematically identical to Curiosity and the VSTB and has a mass of 318 kg. This reduced mass yields a system that weighs approximately the same on Earth as Curiosity does on Mars, enabling realistic mobility tests to be performed both at JPL's Mars Yard and in the field. Much of the physical Traction Control testing was performed using this vehicle, while the VSTB was used primarily for software validation.

Two systems of loads measurement were used throughout testing; a ground-emplaced load cell affixed to the bottom of a 15 cm radius hemispherical dome, and three hub-mounted load cells, which were attached to the center of each of the vehicle's starboard side wheels. The hemispherical dome functioned both as a measurement system and a simplified obstacle, whereby a wheel's ascent and descent of the dome followed a continuous arc rather than a step function. The hemisphere was constructed of aluminum and was anchored to the ground by four 12" stakes, yielding a rigid structure with negligible flexure during traverses. The ATI Omega 160 six axis force-torque sensor was used for both the ground and hub-mounted load cell.

Figure 10 shows an example of the reduction in measured resultant load (about X, Y, and Z axes) of the hub-mounted load cells during a Scarecrow forward traverse of the dome obstacle. The top graph represents a Traction Control *on* run,

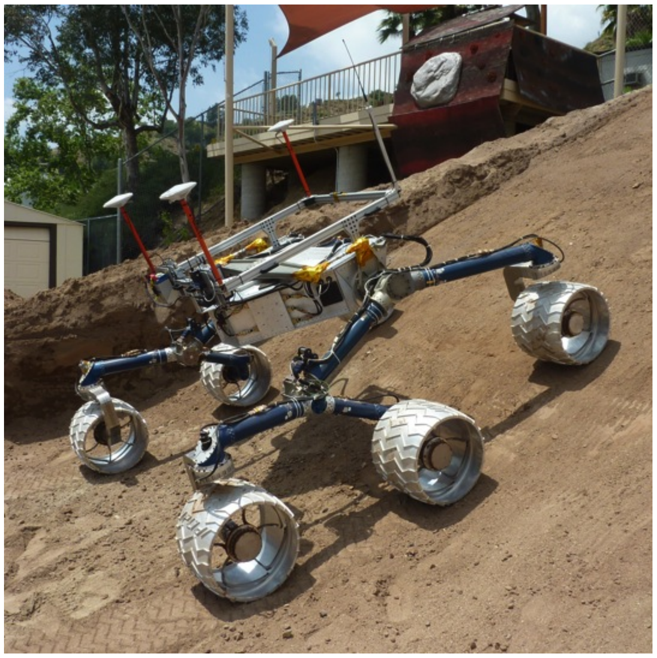


Figure 9. The Scarecrow mobility test system.

while the bottom graph shows the loads with Traction Control *off*. Blue represents the right front (RF) wheel, greed the right mid (RM) and black the right rear (RR). While each wheel encounters the obstacle, the remaining five wheels remain on relatively flat terrain, enabling the clear temporal demarcation of individual wheel contacts. This is useful in both the recreation of events in simulation and also in the teasing out Traction Control behaviors during data analysis. In the provided example, the algorithm reduces the resultant loads on the front wheels, both peak and average. Two other observations are worth noting; 1) the effect of Traction Control reduces the loads on the front and mid wheels more so than the rear wheel, and 2) Traction Control also reduces the extent to which non-obstacle-climbing wheels are unloaded. These points were observed to be true throughout both Scarecrow and VSTB testing.

Point 1 can be explained by considering the geometry of the rocker-bogie suspension system. Under this geometry, both the front and mid wheels are affixed to *leading* suspension arms. That is, the angle from the vertical to the suspension arm is in the direction of travel. A *trailing* wheel, such as the rear wheels (during forward motion) has its suspension arm rotated in the opposite direction. This leads to a case similar to that of a person trying to get a dolly or cart over a curb. If the person pushes (leading wheel) the dolly, the force required to surmount the obstacle is much greater than if the dolly were pulled (trailing wheel). In the leading wheel case, Traction Control is able to modulate the speed of the non-obstacle-climbing wheels such that they do not impart excessive *pushing* loads, and the reduction is wheel loading is highly evident. In the trailing wheel case, the resultant loads are significantly lower due to the geometry of the vehicle, which is turn reduces the noticeable effect of Traction Control. Figure 11 shows similar results for test cases run with the VSTB. Here, similar trends of *leading* wheels seeing significant reductions in loading with Traction Control enabled can be observed.

During development testing of the Traction Control algorithm

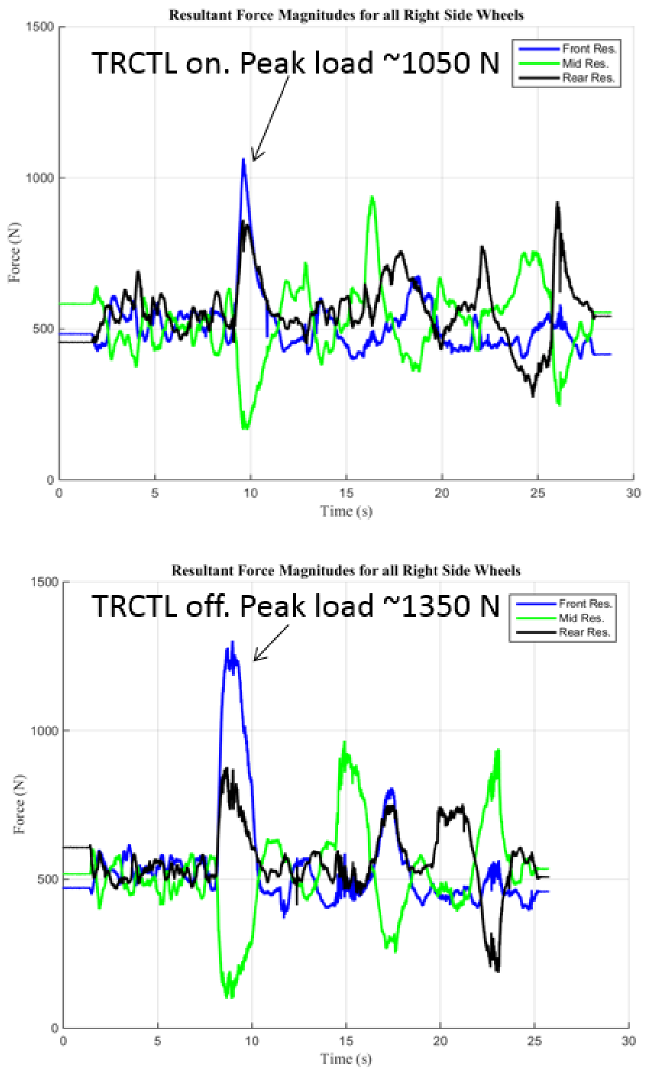


Figure 10. Resultant loads recorded by Scarecrow hub-mounted load cells during a single straight forward drive over the dome with the right wheels. This result is representative of the set of tests of this type.

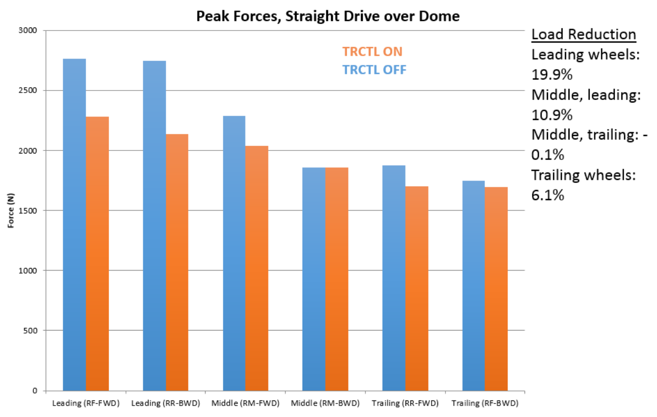


Figure 11. Resultant loads recorded by ground-based load cell during a single straight forward and backward drive over the dome with the VSTB right wheels. This result is representative of the set of tests of this type.

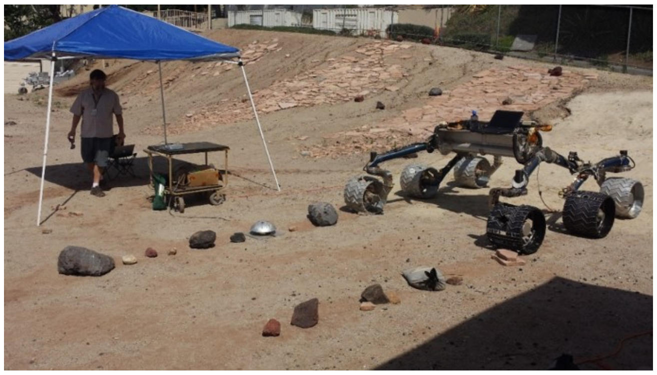


Figure 12. Complex terrain tracks used in Scarecrow development testing. The aluminum dome, which was mounted to the top of a load cell, is one of the objects driven over by the right wheels.

it was calculated that the average reduction in resultant loads on relatively benign terrain was 19% for front leading wheels, while middle leading wheels experienced an 11% reduction.

Complex Terrain Testing

Tests with both the VSTB and Scarecrow rovers were also performed on complex terrain on tracks similar to that shown in Figure 12. Figure 13 shows the reduction in loads during a Traction Control run over the complex, loose terrain. Note that the runs are compared only for the first 90 seconds. After this time, it was observed that the vehicle would consistently drive off track without Traction Control due to inadvertent yaw. This point is addressed in the following subsection.

To compensate for inadvertent yaw, rigid, high friction tiles were used to cement the obstacles into the ground. The Scarecrow rover was commanded to perform a 10 meter arc across the tiles, which were spaced at approximately 1 meter intervals. The tiles were placed such that the obstacles being traversed by either the right or left sides of the vehicle were approximately symmetric (Figure 14). This course was driven three times with Traction Control enabled and three times with Traction Control disabled. Example RR wheel results are shown in Figures 15 and 16. These figures show the cumulative frequency distribution of loads and drive torques during Traction Control enabled and disabled runs. Green represents Traction Control enabled, while red represents disabled. The leftward shift in the CDF curve clearly identifies a reduction in the integral of both histograms, demonstrating a lowering of the forces and required drive torques when using Traction Control.

Following the cemented tile testing, a summary was produced to detail the average reduction of loads and drive torques. As seen throughout development, Traction Control provides a modest yet consistent reduction in wheel loading. Drive actuator torques were also significantly reduced. This summary is provided in Figure 17.

Yaw Reduction Testing

An auxiliary benefit of the Traction Control algorithm is the reduction of unintended yaw during longer drive steps. By reducing the degree to which wheels push/pull each other into obstacles or sand ripples, Traction Control has the effect of balancing loads across its right and left sides. This balancing aids in the reduction of yaw on rigid terrain, and both yaw and wheel slip on sandy terrains. Figure 13 shows the reduction of

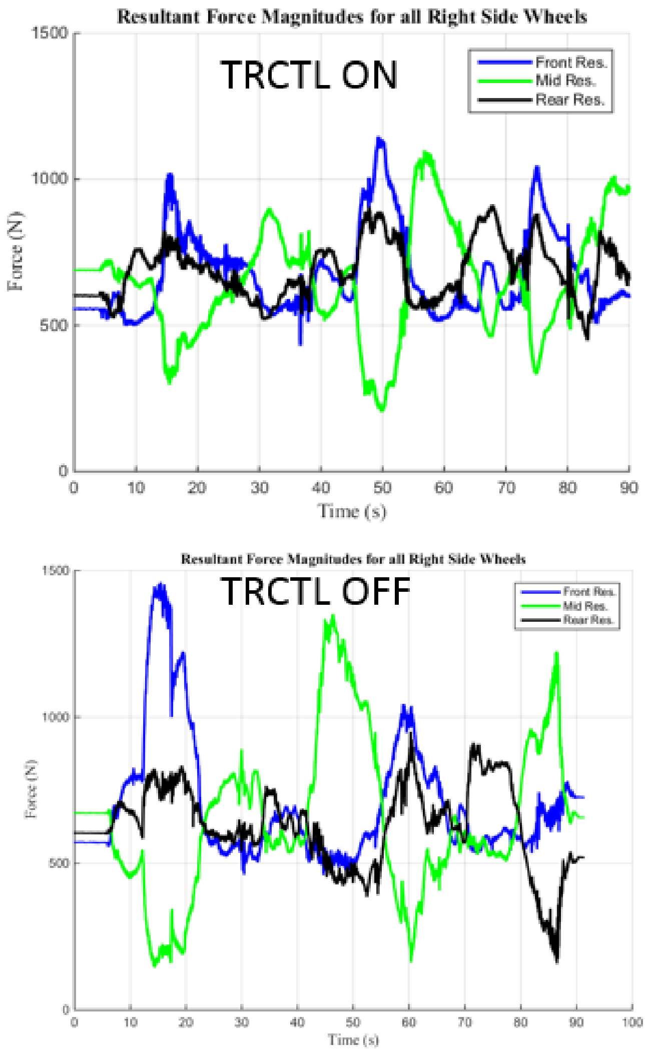


Figure 13. Resultant loads on Scarecrow hub-mounted load cells during a single forward traverse of the complex obstacle track.

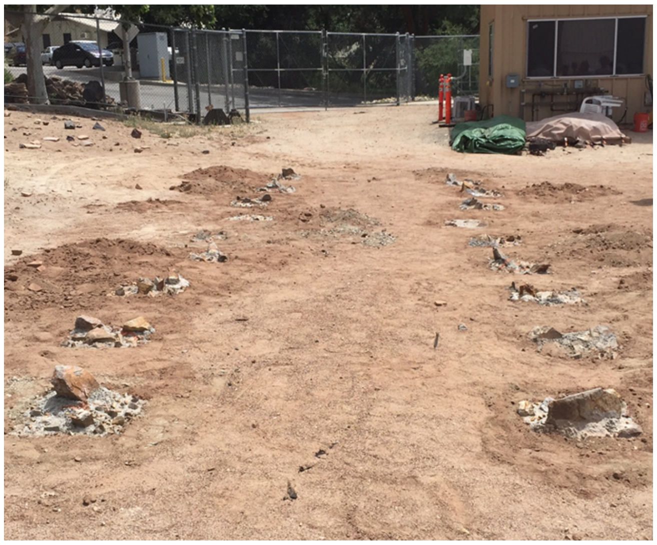


Figure 14. Complex terrain course constructed with high-friction, embedded rock tiles.

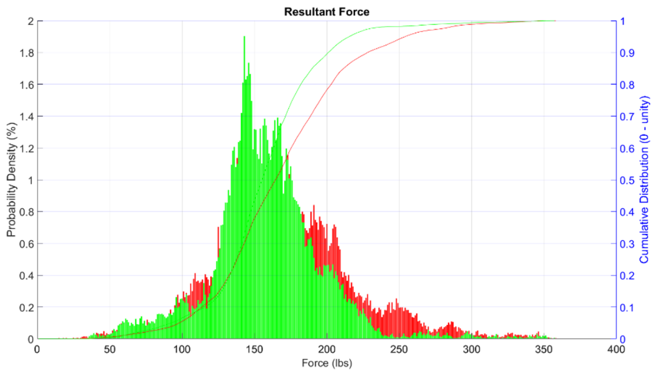


Figure 15. Cumulative frequency distribtion of resultant loads on Scarecrow RR wheel during traverse of high friction, embedded rock tiles. Green and red correspond to Traction Control enabled and disabled, respectively.

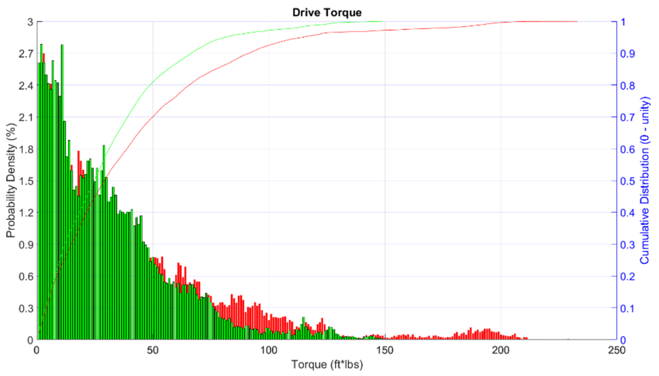
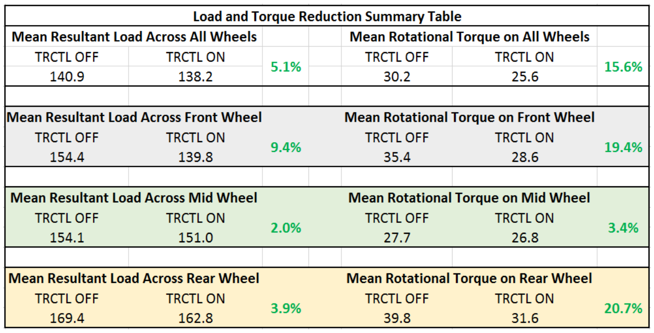


Figure 16. Cumulative distribution of drive torques experienced by Scarecrow RR wheel during traverse of high friction, embedded rock tiles. Green and red correspond to Traction Control enabled and disabled, respectivly.



Load and Torque Reduction Summary Table					
Mean Resultant Load Across All Wheels			Mean Rotational Torque on All Wheels		
TRCTL OFF	TRCTL ON	5.1%	TRCTL OFF	TRCTL ON	15.6%
140.9	138.2		30.2	25.6	
Mean Resultant Load Across Front Wheel			Mean Rotational Torque on Front Wheel		
TRCTL OFF	TRCTL ON	9.4%	TRCTL OFF	TRCTL ON	19.4%
154.4	139.8		35.4	28.6	
Mean Resultant Load Across Mid Wheel			Mean Rotational Torque on Mid Wheel		
TRCTL OFF	TRCTL ON	2.0%	TRCTL OFF	TRCTL ON	3.4%
154.1	151.0		27.7	26.8	
Mean Resultant Load Across Rear Wheel			Mean Rotational Torque on Rear Wheel		
TRCTL OFF	TRCTL ON	3.9%	TRCTL OFF	TRCTL ON	20.7%
169.4	162.8		39.8	31.6	

Figure 17. Summary of the reduction in average load and drive torques over high friction tiles.

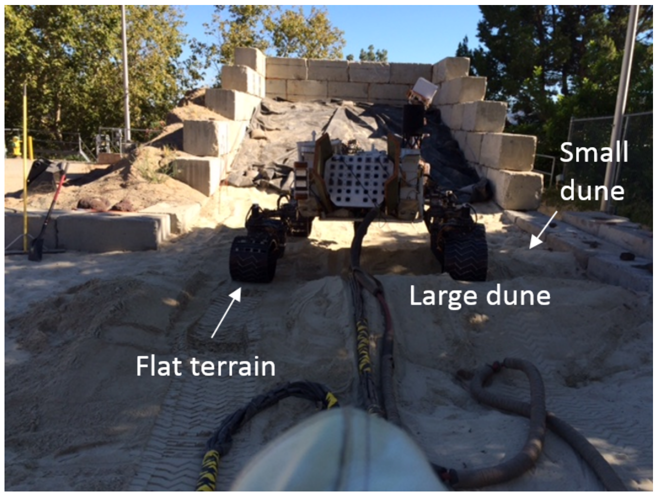


Figure 18. VSTB yaw testing on loose sand.

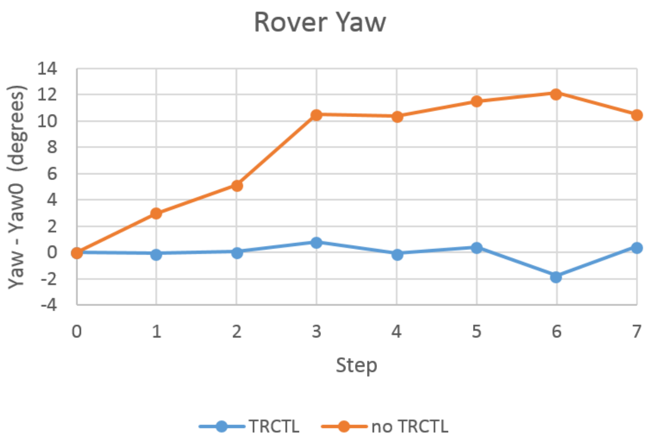


Figure 19. Recorded delta yaw comparison between TRCTL on and off runs over rippled sand.

resultant loads during a traverse of the *complex* terrain, shown in Figure 12. During tests utilizing the VSTB, yaw reduction was also noted while traversing non-symmetric sand ripples, as shown in Figure 18. Ripples having a height of approximately 35 cm resulted in significant inadvertent yaw during non-Traction Control runs, while this was abated following the implementation of Traction Control, as shown in Figure 19. However, ripples with half that height resulted in no significant difference in inadvertant yaw between Traction Control on and off runs.

As evidenced by Figure 19, Traction Control maintains a relatively steady heading while the run with Traction Control disabled resulted in significant delta yaw over a relatively short arc length. While this is an ancilary effect and not one that was sought at the outset of this task, it is believed that this may be seen as a significant advantage offered by Traction Control for future projects such as Mars 2020.

4. Integration Into Mission Operations

The development of Traction Control software was completed long after the most recent update to Curiosity's flight software. Thus it was incorporated into the MSL flight system as a *hot patch* applied to the current (R12) version of flight software. We assessed the impact the new behavior would have on the general planning model (each day's plan predicts power, data, and duration of all activities), as well as the changes imposed on the creation and validation of sequences of drive commands.

Flight Software Integration

MSL has fully updated its onboard flight software (FSW) four times since launch. The most recent full update was the R12 release, which was built in August 2014 (prior to completion of this algorithm) and deployed in January 2015. The R12 release incorporated updates that simplified the later integration of Traction Control. Those updates included: software hooks added to invoke a function pointer to evaluate drive wheel rates at 8 Hz; extra placeholder integer and floating point parameters; a new Data Product Object downlink record type with placeholders for a small number of integer and floating point downlink values; and new commands to enable and disable the future capability.

Once the new software was completed, it was compiled into a single object file following standard protocols for MSL FSW hot patches [5]. In each bootup period that requires it, this file is loaded into the VxWorks operating system and assigned to a function pointer global variable via a shell command script. Traction Control then becomes available for use during the remainder of the current boot cycle.

Incorporating this software as a patch allowed us to take advantage of this capability relatively quickly. MSL project procedures require thorough regression testing and planning across all FSW when updating the whole FSW image, whereas a patch only requires regression test of mobility capabilities. The patch is also hundreds of times smaller than a full FSW load would be, requiring fewer days to uplink to Mars and reducing the impact to the operations schedule.

Mission Planning Integration

Several aspects of operations planning were impacted by the ongoing use of this new software. The MSL ground operations team models the Duration, Power, and Data Volume usage of all onboard activities. Traction Control software is active during nearly all drive modes, so the resource models for all drive activities were updated.

Based on the testing we performed on Earth, we predicted that Traction Control would slow down drives by no more than 25%. Our initial drives on Mars were planned taking this worst-case model into account. But while the actual slowdown is terrain-dependent (the more uneven the terrain, the greater the overall slowdown), in practice we have observed slowdowns only on the order of 10% and have since updated our models to presume no more than 15%. Power modeling was not changed, except for the implication that longer drives would require more CPU and Inertial Measurement Unit (IMU) energy overall: this implementation adds less than 3% to the total CPU usage while driving. But in terms of Data Volume, the extra data generated by the initial implementation was significant; it typically nearly doubles the amount of high-rate data collected due to the records of individual wheel speed commands being collected at 8 Hz.

All MSL drive command sequences are constructed by the Rover Planner team, so their environment was updated to support nominal use of this new capability. Standard startup command sequences were updated to load and enable the software patch, removing the need to explicitly remember to turn it on for each drive. The command sequence static analyzer *RP-check* [6] was updated to issue a warning for any drive commanded without Traction Control being enabled. The Rover Sequencing and Visualization Program (RSVP) [7] Surface Simulation component was updated to accept an average expected Traction Control Speed Ratio (defined below) as input and adjust planned drive durations accordingly. And strategic downlink analysis tools were updated to enable Mobility Downlink and Rover Planner team members to quickly measure the actual slowdown seen over any number of recent drives, so they could apply it to the duration estimate needed for the current drive.

Downlink Inputs to Tactical Planning

Traction Control software generates downlink telemetry to enable ground understanding of its performance. Section 5 will cover our detailed downlink assessment capabilities, and Table 1 shows a list of all the values available for daily and long-term trending queries. These fields are populated within the MSL Strategic Mobility database, an update to the comparable system used for the Mars Exploration Rover Spirit and Opportunity missions [8].

For planning purposes, the primary number of interest is the Traction Control Speed Ratio, defined as follows. For any given 8 Hz sample, we know the speed at which each wheel is commanded to drive. We can also compute the equivalent speed that would have been commanded by the planar (double Ackermann) style non-Traction Control driving algorithm. To estimate the overall progress of the vehicle, we compute the ratio of commanded speed / Ackermann speed for each wheel, disregard the slowest wheel (which might be commanded at 0 m/s if it rests on the center of the turning circle) then pick the median of the five remaining values. Figure 20 shows this Traction Control Speed Ratio evaluated at 8 Hz during drive operations in 2545 samples from Sol 1814.

Rover Planners use the average Speed Ratio over any given range of sols to inform their duration estimate for the next drive. Overall, the average Speed Ratio during the first six months of operations from sol 1646 through 1822 is 0.899 from 351992 samples. That is, Traction Control tends to drive 10% slower than planar drives in the current terrain.

However, the net impact on the whole system has been smaller. MSL typically uses onboard image processing to refine its position estimate, using a Visual Odometry algorithm [9], [10]. MSL typically drives long distances in 1 meter steps, stopping after each step to acquire new Visual Odometry images and process them. The time needed to stop and process images is nearly double the time needed to drive the 1 meter step, which means the time spent physically moving is less than 40% of the total drive activity; hence the slowdown impact of Traction Control on Visual Odometry-enabled drives is less than 4% on average.

5. Assessment Tools

Motion history data products are generated whenever rover mobility is commanded. Since R12, these data products record a subset of the estimates and control signals computed by the Traction Control algorithm, when it is enabled. A suite of analysis tools was developed to process mobility data products and assess the performance of Traction Control in both the ground testing and flight phases. Per Figure 5, of particular interest are time-series plots of the following quantities: rover attitude angles and rates, suspension (bogie and differential) angles and calculated rates, estimated contact angles for each wheel, and commanded angular velocity for

Field	Description	Units
f_trctl_active	Active	Boolean
f_trctl_algorithm	Algorithm	0_ADAPTIVE, 1_PLANAR
f_trctl_control_mode	Mode	0_RATE, 1_POS
f_trctl_telem_mode	Orientation Telemetry (otlm) Mode	0_WORLD, 1_ROVER
f_trctl_speed_ratio	Ratio of Commanded to Ackermann Reference Speed	Float between [0, 1]
f_differential_rate	Differential Angular Velocity	radians/sec
f_bogie_l_rate	Bogie L Angular Velocity	radians/sec
f_bogie_r_rate	Bogie R Angular Velocity	radians/sec
f_wheelie_lm	LM Wheelie Correction Active	Boolean
f_wheelie_lr	LR Wheelie Correction Active	Boolean
f_wheelie_rm	RM Wheelie Correction Active	Boolean
f_wheelie_rr	RR Wheelie Correction Active	Boolean
f_world_x_rate	Local Level Frame X Rate	radians/sec
f_world_y_rate	Local Level Frame Y Rate	radians/sec
f_world_z_rate	Local Level Frame Z Rate	radians/sec
f_rover_x_rate	Rover Frame X Rate	radians/sec
f_rover_y_rate	Rover Frame Y Rate	radians/sec
f_rover_z_rate	Rover Frame Z Rate	radians/sec
Fields that are repeated for each wheel: Left/Right, Front/Middle/Rear		
f_contact_angle_(lf, lm, lr, rf, rm, rr)	Wheel Contact Angle	radians
f_theta_dot_cmd_(lf, lm, lr, rf, rm, rr)	Wheel Commanded Rotational Speed	radians/sec
f_wheel_rate_cmd_(lf, lm, lr, rf, rm, rr)	Wheel Commanded Linear Speed	cm/sec
f_wheel_rate_ack_(lf, lm, lr, rf, rm, rr)	Ackermann Wheel Reference Linear Speed	cm/sec
f_contact_fail_(lf, lm, lr, rf, rm, rr)	Wheel Contact Angle Computation Failed	Boolean
f_rate_fail_(lf, lm, lr, rf, rm, rr)	Wheel Rate Computation Failed	Boolean

Table 1. Strategic Database names for high-rate Traction Control downlink data (collected at 8 Hz)

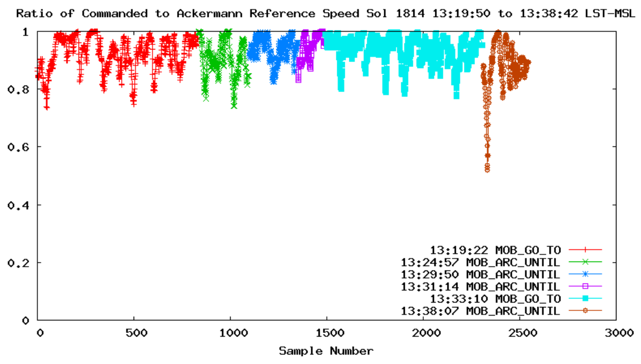


Figure 20. The average Traction Control Speed Ratio for Sol 1814 is 0.924 from 2545 samples.

each drive actuator. Where possible, the magnitude axes were pinned for informative comparison across all drives.

During ground testing, the quality of the contact angle estimates was assessed relative to both the configured obstacle environment and the commanded motion. For example, driving on flat terrain should result in small and zero-mean estimated contact angles, whereas the front left wheel driving over a hemisphere should (all else being equal) result in a clear signature for that wheel only. In flight, a comparison is possible only with respect to a computed mesh, which has its own error sources; however, any unrealistic contact angle estimates that differ significantly from terrain models would be observable. The commanded angular velocities, which are the Traction Control outputs, can be assessed for consistency with the joint ego-motion estimate, including contact angles, attitude angles and rates, and suspension angles and rates. The juxtaposition of the contact angle and angular rate plots was crucial for algorithm assessment.

Also analyzed were time-series plots for the current estimated by the motor controller flight software to be supplied to each drive actuator in response to the wheel angular rates commanded by Traction Control. These plots were used to verify that Traction Control did not significantly increase the current demanded by the drive actuators during mobility activities. For each drive actuator, the current, voltage, and wheel contact angle estimates were also used to plot modeled torque on each wheel. Because torque is indirectly comanded via wheel rate, correspondence between the shape of the contact angle estimate curves and the drive torque curves would be a measure of the efficiency of the algorithm when scaling obstacles and is the subject of ongoing investigation.

After free-floating wheelies on the middle wheels were induced in some test configurations, wheelie detection and suppression logic was added, with indicator variables for the wheelie statuses being exposed in telemetry. The combination of these time-series plots was helpful in formulating and verifying the wheelie detection and suppression strategies.

6. Performance In Flight

Three checkout tests were performed on Mars after software delivery was approved in March 2017. For checkout test 1, performed on Sol 1644, the software patch was uploaded to Curiosity and initial parameters were set, saved in non-volatile memory, and recorded in a drive module data product. After the drive module data product was downlinked, the operations team verified the parameters were correctly saved on-board the rover. For checkout test 2, performed on Sol

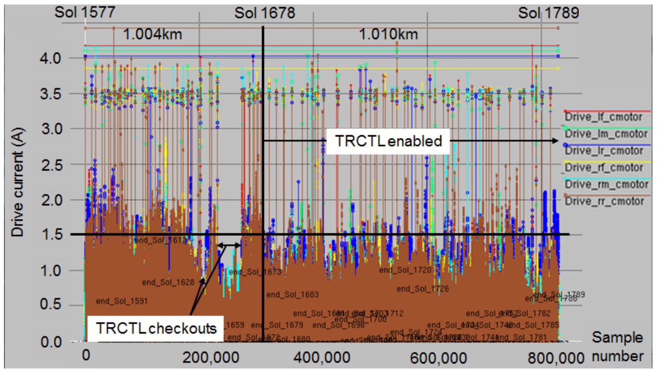


Figure 21. Raw drive currents 1 kilometer before and after nominal use of Traction Control. For a qualitative comparison of the data before and after Sol 1678, a horizontal reference line is drawn at 1.5A. This graph illustrates one benefit of the new approach is lower peak drive currents, but one cost is longer drive times.

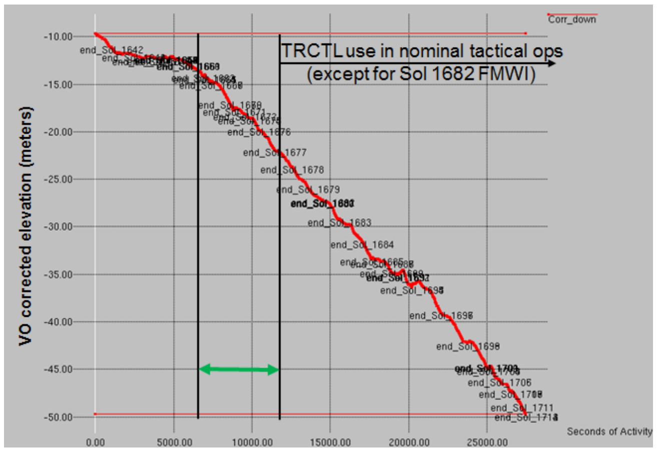


Figure 22. The rover elevation rate of change is similar for a period before starting nominal use of Traction Control (as denoted by the green arrow) and for a period afterward. Since positive z is downward, the rover was driving uphill.

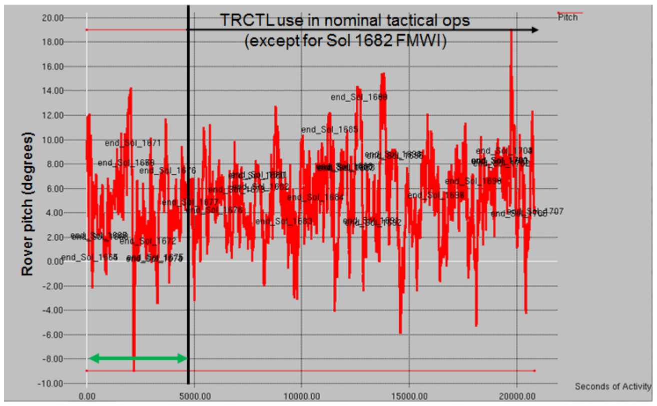


Figure 23. The rover pitch during the selected Traction Control disabled and enabled data sets are similar.

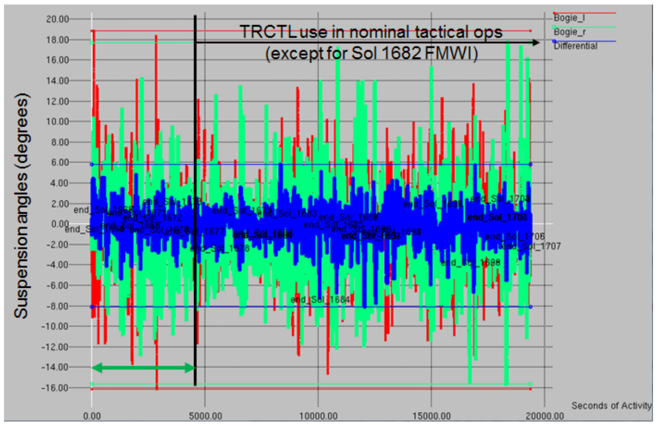


Figure 24. The rover suspension range of motion during the selected Traction Control disabled and enabled data sets are similar.

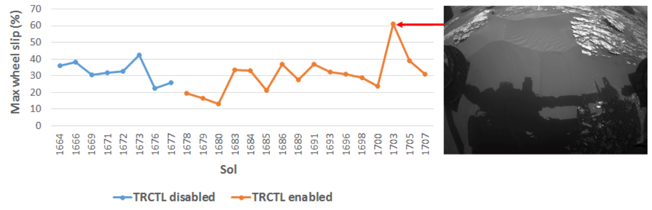


Figure 25. Max rover slip for the drives in the Traction Control disabled and enabled data sets. The Sol 1703 drive faulted due to excessive slip while driving uphill on sand. The average max rover slip was 2.1% lower for the Traction Control enabled data set.

1646, a short (5 meter) drive was executed with Traction Control enabled, logging drive telemetry at 64Hz. After on-console assessment of the telemetry by the operations team, the activity lead approved proceeding to checkout test 3. For checkout test 3, performed on Sol 1662, a 20 meter drive was executed with Traction Control enabled, again logging drive telemetry at 64Hz. Following a review of checkout test telemetry with MSL management, the new capability was approved for nominal use on Curiosity in April 2017.

Since its first nominal use on Sol 1678, Curiosity has driven 1.450 kilometers in 68 drives, as of October 6, 2017, almost all of it with Traction Control enabled. The exceptions are three 1.2-meter drive segments on Sols 1682, 1730, and 1798 to perform full MAHLI wheel imaging (FMWI), and 2.5 meters at the beginning of the Sol 1787 drive when recovering from a drive fault on the previous sol that left the right bogie close to its soft limit. FMWI is currently performed every 500 meters to assess changes in damage to the wheels. Since it consists of rotating the wheels to five equidistant positions for imaging, Traction Control is disabled to ensure that each wheel turns the same amount.

Since there are no sensors on Curiosity to measure the loads on wheels, the operations team monitors wheel currents as an indicator of how much work each wheel is doing. Figure 21 shows the raw drive currents over a 2 kilometer period; 1 kilometer before and after nominal use of Traction Control began on Sol 1678. The terrain traversed using Traction Control had a 1.7x higher average uphill slope: on Sol 1678 Curiosity's elevation had increased by 42.61 meters over the last kilometer, but gained 69.49 meters over the next

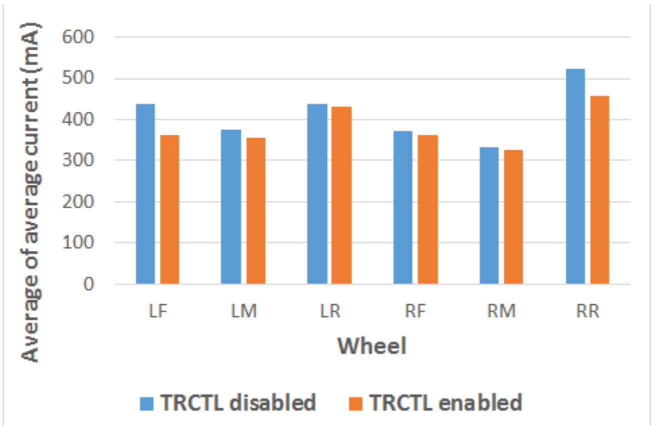


Figure 26. Average of average drive currents for the Traction Control disabled and enabled data sets. The average of average wheel current for all wheels was lower for the enabled data set.

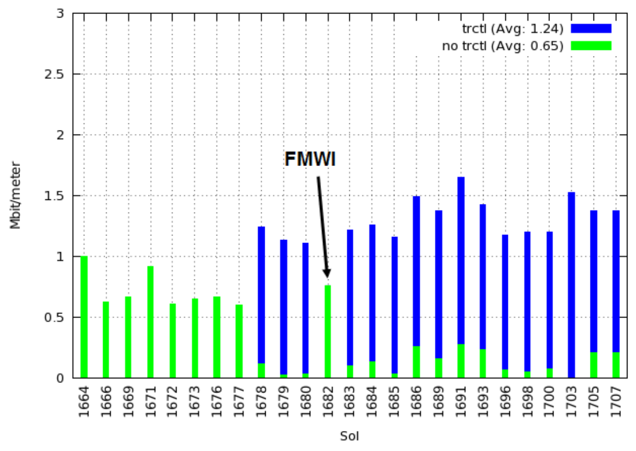


Figure 27. Compressed motion history data product size for the Traction Control disabled and enabled data set. On average, the motion history data product size was 1.91x larger when the new capability is enabled.

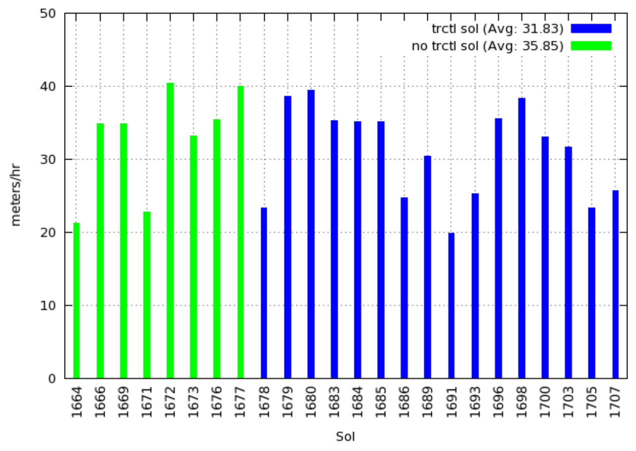


Figure 28. Drive traverse rate for the Traction Control disabled and enabled data sets. On average, the enabled drives took 11.2% longer. The Sol 1682 FMWI drive was not included in this analysis since wheel imaging is performed after every drive motion.

kilometer. The spikes above 3.0A are mostly from open-loop startup current at the beginning of each drive. The lower drive currents are in-part due to driving on less-complex terrain during the checkout tests. The mean of the mean current for each drive step over all wheels for the 1 kilometer before and after Traction Control was enabled for nominal use on Sol 1678 was 350 mA and 321 mA. The mean of the max current for each drive step over all wheels for the 1 kilometer before and after Sol 1678 was 368 mA and 345 mA. (The Traction Control checkout data were excluded when generating those statistics). Lower peak and average drive currents with Traction Control enabled are consistent with the development and V&V test results. The reason there is more data to the right of Sol 1678 is that Traction-Control-enabled drives take slightly longer to complete.

One limitation of comparing drive currents from before and after starting nominal use of Traction Control is the drives are not over the exact same terrain. It could be that some of the terrain is easier to traverse, for example, downhill vs. uphill. To address this, a Traction Control disabled set of sols (1664-1677) and Traction Control enabled set of sols (1678-1707) were selected for analysis which were close in proximity and had similar rover pitch and suspension angle profiles. Figure 22 contains a plot of the rover elevation after being corrected by Visual Odometry. There are 8 uphill drives between Sols 1664 -1677 that have a similar elevation rate of change as 17 drives between Sols 1678-1707.

Figures 23 and 24 contain rover pitch and suspension angles for the two ranges of sols selected for analysis. The average rover pitch is 0.7 degrees higher for the Traction Control enabled data set. In the suspension angle plot, while the average absolute value of the left bogie is 0.30 degrees higher for the disabled data set, the average absolute value of the right bogie and differential are 0.28 and 0.11 degrees higher for the enabled data set. That these values are small indicates that the terrain difficulty for the two data sets is similar.

Figure 25 contains a plot of the max rover slip for the drives in the two data sets. The Sol 1703 drive faulted after 2.88 meters when the rover experienced wheel slip of 61.2%, exceeding the default 60% slip limit. The front hazard camera image in Figure 25 illustrates that Curiosity was driving uphill through sandy terrain at a rover pitch of 9.5 degrees when the drive faulted. The average max rover slip is 2.1% lower for the enabled data set. Since there were no comparable excessive slip drives in the disabled data set, this drive was removed from the Traction Control enabled data set as an outlier when comparing drive currents between the two data sets. In addition, the data for the Sol 1682 FMWI drive was removed from the enabled data set, since the capability is disabled for FMWI drives. The total odometry for the 8 drives in the disabled data set was 135.7 meters. The total odometry for the 15 drives left in the enabled data set was 389.7 meters.

By default, drive telemetry is recorded at 8Hz for both Traction Control disabled and enabled drives. For both the disabled and enabled data sets, the raw non-startup drive current for each wheel was averaged over each drive. Then the average drive current for each wheel was averaged over all the disabled and enabled drives. Figure 26 contains a graph of the average of average drive current for each wheel for both data sets. The average of average wheel current for all wheels was lower for the enabled data set, most notably for the LF and RR wheels. The LF, LM, LR, RF, RM, and RR averages of average wheel current were 17.0%, 5.2%, 1.5%, 2.3%, 2.1%, and 12.9% lower for the enabled data set.

When Traction Control is enabled, a timeout is calculated for each drive step. Exceeding this timeout is the only new fault type introduced. Of the 66 nominal-use drives thus far, only one has ended early due to such a timeout fault; the Sol 1786 drive faulted after 15.86 meters of the planned 27.9 meters when a 32.77 second timeout was exceeded while the right rear wheel was driving over a large rock. Incidentally, the right bogie suspension angle was 0.3 degrees away from exceeding its 18 degree limit; the drive was seconds away from being stopped with a suspension fault.

During development testing, a middle wheel wheelie behavior was observed on high-friction terrain. To prevent propagation of a middle or rear wheelie with Traction Control enabled, a wheelie suppression behavior, which adjusts the speed of the other bogie wheel to lower the elevated wheel, was added to the Traction Control software. The detection of a wheelie event occurs when the suspension rate and bogie angle exceed a threshold, and the motor current magnitude is below a threshold. The amount a bogie wheel is adjusted is proportional to the bogie angle and bogie angle rate. These values are parameterized and set conservatively. When driving over complex terrain with Traction Control enabled, it is not unusual for the wheelie detector to be triggered for short durations. The average distance between wheelie detections that occurred in the Traction Control enabled Mars data set was 2.4 meters (with average duration 2.5 seconds), consistent with the 1.7 meter average distance between wheelie detections observed during the Earth-based testing on the complex terrain in Figure 14.

Two of the costs of Traction Control are larger motion history data product size and longer traverse times. The compressed motion history data product size per meter is shown in Figure 27 for all drives in the two data sets. Note that there is usually at least a small portion of Traction Control disabled motion history data products even on drives where Traction Control was enabled. This is because turn-in-place motions and commanded arcs of less than 10cm are not performed using this algorithm. The compressed enabled motion history data products were on average 1.91x larger than the compressed disabled motion history data products. This is lower than the worst case predict from VSTB measurements on complex terrain (2.2x), higher than the best case predict from the VSTB measurements on flat terrain (1.8x), and close to what is currently modeled (1.9x).

For the two data sets, the time between the dispatch of first motion command and completion of last motion command for each drive was obtained from the drive telemetry. For each drive, the total Visual Odometry corrected odometry divided by this time was used to generate the drive traverse rates in Figure 28. For the two data sets, the Traction Control enabled drives took on average 11.2% longer than the Traction Control disabled drives.

7. Summary

The Traction Control algorithm presented in this paper was developed in response to the increased Curiosity rover wheel damage rate observed in October 2013. Based upon the MSL project investigation into the causes of wheel damage, reducing the forces imparted on individual wheels while climbing sharp, embedded rocks may result in extending wheel life. This Traction Control algorithm departs from the previously used Ackermann steering model, which assumes level terrain. The algorithm merges realtime data from the rocker-bogie suspension system and IMU data to estimate wheel contact points with the terrain, and commands speeds based on the climbing behavior of each individual wheel.

The algorithm was implemented as a hot patch to the current testbed version of flight software, rather than being released as a new version of flight software. This decision allowed for a faster implementation, because the patch size was less than 0.5% the size of a new version of flight software, and required only regression testing of mobility and fault protection capabilities. During V&V testing of the Traction Control algorithm, all mobility commands executed nominally while running the software patch. These comprehensive ground tests on the flight-like rover testbed included testing on mixed terrain types and various rock heights; the results presented here demonstrate modest reductions in the wheel loading. The average reduction in resultant loads on relatively benign terrain was 19% for leading front wheels, while leading middle wheels experienced an 11% reduction. Smaller, but consistent, reductions in wheel loads and drive torques were seen on more complex terrain. Development and V&V testing also demonstrated the ancillary benefits of reduced unintended yaw and rover slip.

The MSL project approved the Traction Control flight software patch for nominal use in flight in April 2017. Curiosity's first drive using Traction Control successfully occurred on Sol 1646. Initial performance results from flight data shows that drives using the Traction Control software result in lower peak and average drive actuator current. In comparing a Traction Control-enabled set of sols (1678-1707) with a Traction Control-disabled set of sols (1664-1677), the average reduction in wheel currents was 10% for the front wheels and 7% for the rear wheels. While we cannot directly measure wheel loads or torques seen in flight, these findings suggest that the wheels are experiencing lower forces. Additionally, the new algorithm has resulted in a lower maximum rover slip (average 2.1% lower) over these drives. Overall, the primary costs of the Traction Control software (11.2% longer traverse time and 1.91x larger Mobility data products) are outweighed by the benefit to MSL of reduced forces on the wheels. Although it is difficult to predict how much the wheel force reduction will slow the rate of wheel wear, MSL is planning trending of rover performance and wheel state to understand the effects of Traction Control on wheel damage. Based on preliminary flight validation, the Traction Control algorithm is performing as expected.

Future Work

The performance of the Traction Control software will continue to be assessed on Mars. Additional tools to aid in the trending of flight data and to determine the efficacy of Traction Control are currently in development. In addition, improvements to the Traction Control software are currently being discussed. A version which incorporates torque feedback into wheel speed commanding to better achieve desired torque at each wheel as a function of contact angle has been proposed. This modification is hypothesized to further reduce the resultant contact force on each wheel, although this potential improvement in performance has not yet been quantified. This algorithm is currently in development, and the discussion of its implementation is ongoing.

Future rover missions are also evaluating Traction Control for possible use, largely in part for the potential reduced yaw error and reduced slip benefits demonstrated here.

Appendix

Table 2 below defines the key variables and parameters referenced in this paper.

Table 2. Description of key symbols

Symbol	Description
${}^{b}R_a$	rotation from frame a to frame b
${}^{a}\vec{AB}$	vector $\vec{AB}$ expressed in frame a
${}^{a}v_A$	velocity of point A expressed in frame a (relative to inertial frame)
O	rover origin point
D	rocker pivot point
B_1	left bogie pivot point
B_2	right bogie pivot point
A_i	center of wheel i
l_{fd}	length between front wheel and rocker in body x-z plane
l_{db}	length between rocker and bogie in body x-z plane
l_{bm}	length between bogie and middle wheel in body x-z plane
l_{br}	length between bogie and rear wheel in body x-z plane
κ_1	angle between $\vec{DA_1}$ and body x axis on flat ground
κ_2	angle between body x axis and $\vec{B_1D}$ on flat ground
κ_3	angle between $\vec{B_1A_3}$ and body x axis on flat ground
κ_4	angle between body x axis and $\vec{A_5B_1}$ on flat ground
R_w	wheel radius
x_{fm}	longitudinal distance between front and middle wheels on flat ground
x_{mr}	longitudinal distance between middle and rear wheels on flat ground
y_{of}	lateral distance between origin and front wheels on flat ground
y_{om}	lateral distance between origin and middle wheels on flat ground
y_{or}	lateral distance between origin and rear wheels on flat ground
x_{od}	longitudinal distance between rover origin and rocker on flat ground
z_{od}	signed vertical distance between rover origin and rocker on flat ground
r	turn radius (distance between rover origin and center of rotation)
$\dot{x}$, $\dot{y}$, $\dot{z}$	linear velocity of rover origin along x, y, z body axes
ϕ	rover roll angle
θ	rover pitch angle
ψ	rover yaw angle
$\omega_{x,y,z}$	rover angular rates along body x, y, z axes (relative to inertial frame)
β	left rocker angle
ρ_1	left bogie angle
ρ_2	right bogie angle
η_i	contact angle of wheel i
ψ_i	steering angle of wheel i
$\dot{\theta}_i$	angular rate of wheel i
${}^{y}\zeta_i$	angular rate of drive actuator for wheel i along its y axis

Acknowledgments

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The authors would like to thank the Mars Science Laboratory Program for supporting this research.

References

[1] R. Arvidson, P. DeGrosse, J. Grotzinger, M. Heverly, J. Shechet, S. Moreland, M. Newby, N. Stein, A. Steffy, F. Zhou, A. Zastrow, A. Vasavada, A. Fraeman, and E. Stilly, "Relating geologic units and mobility system kinematics contributing to curiosity wheel damage at gale crater, mars," *Journal of Terramechanics*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022489816300891

[2] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," vol. 5, Big Sky, Montana, USA, Mar. 2002, pp. 2025–2036.

[3] M. Maimone, A. Johnson, Y. Cheng, R. Willson, and L. Matthies, *Springer Tracts in Advanced Robotics*. Springer Berlin / Heidelberg, 2006, vol. 21, ch. Autonomous Navigation Results from the Mars Exploration Rover (MER) Mission, pp. 3–13. [Online]. Available: http://www.springerlink.com/content/j2827230m1q27717/?p=01b0d14671f64b06b62f7837121bd343&pi=5

[4] K. Iagnemma and S. Dubowsky, "Traction control of wheeled robotic vehicles in rough terrain with application to planetary rovers," *The International Journal of Robotics Research*, pp. 1029 – 1040, 2004.

[5] E. Benowitz and M. Maimone, "Patching flight software on Mars," in *Workshop on Spacecraft Flight Software*, Laurel, MD, USA, Oct. 2015.

[6] M. Maimone, S. Maxwell, J. Biesiadecki, and S. Algermissen, "RP-check: An architecture for spaceflight command sequence validation," in *IEEE Aerospace Conference*, Big Sky, Montana, US, Mar. 2018.

[7] J. Wright, F. Hartman, B. Cooper, S. Maxwell, J. Yen, and J. Morrison, "Driving on mars with RSVP," *IEEE Robotics and Automation Special Issue (MER)*, pp. 37 – 45, 2006.

[8] J. J. Biesiadecki, R. Liebersbach, and M. W. Maimone, "Mars exploration rover mobility and IDD downlink analysis tools," in *International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, Los Angeles, California, USA, Feb. 2008.

[9] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars Exploration Rovers," *Journal of Field Robotics*, vol. 24, pp. 169–186, 2007.

[10] A. Johnson, S. Goldberg, Y. Cheng, and L. Matthies, "Robust and efficient stereo feature tracking for visual odometry," in *International Conference on Robotics and Automation*, Pasadena, CA, USA, May 2008.

Biography

Olivier Toupet *received his M.S. degree in Aeronautics and Astronautics from MIT in 2006. He is currently a research technologist in the Robotic Mobility group at the Jet Propulsion Laboratory. His current research interests revolve around the development of innovative technologies for the Mars rovers. In addition to designing the Traction Control algorithm presented in this paper, for which he was awarded the NASA Exceptional Engineering Achievement Medal, Mr. Toupet is also leading the development of the next generation path-planner that will enable the Mars 2020 rover to navigate its environment autonomously.*

Jeffrey Biesiadecki *is the Principal Investigator of the MSL Traction Control effort. He has been a software engineer at NASA's Jet Propulsion Laboratory since 1993, after completing his Master's degree in Computer Science at the University of Illinois, Urbana-Champaign. He designed and implemented the core motor control and non-autonomous mobility flight software for the Mars Exploration Rovers (MER) and MSL, and was a rover driver for MER and MSL, responsible for command sequences that tell the rover where to drive and how to operate its robotic arm on the surface of Mars. He is presently leading the development of the Mars 2020 rover Sampling and Caching Subsystem flight software.*

Arturo Rankin *received his Ph.D. in mechanical engineering at the University of Florida in 1997. He is currently a Robotic Systems Engineer in the Robot Operations group at the Jet Propulsion Laboratory. During Traction Control development and V&V testing, he led the test effort, for which he received an individual JPL Voyager award. He currently is the MSL Mobility/Mechanisms Downlink team lead, the MSL Flight Software team lead, and a member of the MER Mobility/Instrument Deployment Device Downlink team.*

Amanda Steffy *is a systems engineer in the Flight Engineering group at the Jet Propulsion Laboratory. As part of the MSL Wheel Wear Tiger Team, she conducted Mars Yard testing to understand the wheel damage, led the flight assessment of wheel damage, and developed the strategic wheel wear management plan, for which she received an individual award. Currently, Ms. Steffy is a systems engineer on the Multi-Angle Imager for Aerosols (MAIA) instrument project. She holds a B.S. from Cornell University.*

Gareth Meirion-Griffith *is a Robotics Systems Engineer in the Robotic Mobility Group. Gareth holds a PhD in Mechanical and Aerospace Engineering from the Illinois Institute of Technology (2012), an M.Sc. in Satellite Engineering from the University of Surrey (2008) and a B.Eng. in Aerospace Engineering from Kingston University (2006). He worked in geostationary spacecraft operations for EUMETSAT as part of the Meteosat program from 2006 - 2007 and in automation consulting from 2012 - 2014. Gareth joined JPL in 2015 and is currently a Curiosity rover driver and researcher in the field of predictive mobility models for small-wheeled vehicles operating on off-road, granular materials. He is particularly interested in surface mobility capabilities for the exploration of Ocean Worlds.*

Dan Levine *is a research technologist in the Robotic Estimation and Controls group at the Jet Propulsion Laboratory. His interests are at the intersection of computational statistics and robotics, particularly estimation, perception, mobility, and adaptive sampling. He holds an S.B., S.M., and Ph.D. from MIT's Department of Aeronautics and Astronautics.*

Maximilian Schadegg *received his M.S. degree in Aerospace Engineering from The University of Texas at Austin in 2013. He is currently an engineer in the Inner Planet Navigation group at the Jet Propulsion Laboratory. In addition, he leads the mission design chair for JPL's Advanced Concepts Team (Team X) and is a member of MSL's mobility team. His research interests span many aspects of robotic interplanetary missions, including deep-space navigation, trajectory design, concurrent systems engineering, operations automation, and in situ observations.*

Mark Maimone *is a Robotic Systems Engineer in the Robotic Mobility group at the Jet Propulsion Laboratory. Mark designed and implemented the GESTALT self-driving surface navigation Flight Software for MER and MSL missions; during MSL operations served as Deputy Lead Rover Planner, Lead Mobility Rover Planner and Flight Software Lead; developed downlink automation tools for MER and MSL; and is now a member of the Mars 2020 Rover FSW development team. He holds a Ph.D. in Computer Science from Carnegie Mellon University.*