Modular Dynamic Simulation of Wheeled Mobile Robots

Neal Seegmiller and Alonzo Kelly

Abstract This paper presents a modular method for 3D dynamic simulation of wheeled mobile robots (WMRs). Our method extends efficient dynamics algorithms based on spatial vector algebra to accommodate any articulated WMR configuration. In contrast to some alternatives, our method also supports complex, nonlinear wheel-ground contact models. Instead of directly adding contact forces, we solve for them in a novel differential algebraic equation (DAE) formulation. To make this possible we resolve issues of nonlinearity and overconstraint. We demonstrate our method's flexibility and speed through simulations of two state-of-the-art WMR platforms and wheel-ground contact models. Simulation accuracy is verified in a physical experiment.

1 Introduction

This paper presents a modular method for 3D dynamic simulation of wheeled mobile robots (WMRs). Here, "dynamic simulation" means a second-order physics-based motion model is used, which accounts for inertia and applied forces. This is an expansion on our previously published work on first-order velocity kinematic motion models [15]. "Modular" means that the method accommodates any articulated WMR configuration and any wheel-ground contact model expressed as a function with the specified inputs.

Our colleagues have recently made progress in model-predictive planning [10][16]. To perform well, these planners require accurate motion models that correctly account for wheel slip, rollover, actuator limits, etc. that can also be simulated much faster than real time. Our method strikes a favorable balance between these opposing criteria. We extend efficient dynamics algorithms originally developed for manipulators to account for a non-fixed base and the enforcement of wheel-ground

All authors
Carnegie Mellon University, Robotics Institute, e-mail: {nseegmiller,alonzo}@cmu.edu

contact models. These models specify the nonlinear relationship between force and slip at the wheel-ground interface. The simplest way to incorporate these models is to compute contact forces once per time step based on current slip values, and directly add them at each wheel [1][19][11]. Then the dynamics equations are simply a system of ordinary differential equations; however, the system will be "stiff" requiring a solver that takes very small or adaptive time steps [1]. In contrast, we enforce wheel-ground contact models in a novel differential algebraic equation (DAE) formulation using Lagrange multipliers (Section 3.4). This requires the resolution of nonlinearity and overconstraint issues, but proves to be more stable and efficient.

While our contribution is theoretical, it has immediate practical applications. Using our method a wealth of literature on experimentally derived wheel-ground contact models can be readily applied to full vehicle simulations for planning, control, and estimation purposes. We demonstrate this through simulations of two state-of-the-art WMR platforms and wheel-ground contact models in Section 4.

2 Related Work

Modeling wheel-ground contact is still an active research area. For example, in robotics literature there are terramechanics-based models for rigid wheels in loose soil [11][5][14]. In automotive literature there are models for pneumatic tires [3]. While the ultimate objective of these wheel-level contact models is to improve vehicle-level simulations, few of these publications attempt to do so, perhaps because the available resources for simulation are unsuitable.

Some commercial resources exist for vehicle simulation such as Adams/Car and CarSim. These use very detailed models to the level of small parts and subsystems (e.g. engine mounts); as a result these require the knowledge of many parameters and can be slow. More importantly model configurations are limited to on-road vehicles like cars and trucks. JPL developed Rover Analysis, Modeling, and Simulation (ROAMS) software to model the full dynamics of the Mars Exploration Rovers, including wheel/soil slippage/sinkage interaction [13]. While ROAMS was helpful for design evaluation, it was not feasible for use in motion planning.

Physics-based models that account for wheel slip, rollover, actuator limits, etc. could greatly benefit WMR motion planning, but because implementation and computation costs are high they are seldom used. Ishigami et al. proposed a rover planning algorithm that performs complete dynamic simulation candidate paths, but cited computational cost issues; evaluating just 4 paths (that would each take the rover about 3 minutes to execute) took 47 minutes [12]. Eathakota et al. approximate WMR dynamics in their RRT-like planner to ensure paths satisfy quasi-static and friction cone constraints [7]. Muir and Tian et al. modeled WMR physics in 2D for feedback control purposes [20][23].

Due to its ease of use, Open Dynamics Engine (ODE) is often used to simulate WMRs [11][17][6][15], but it has limited options for wheel-ground contact modeling. ODE can only enforce a rudimentary contact model with Coulomb force limits

approximated by a friction pyramid and slip velocities linearly proportional to force. Our method supports a simulator with ODE's ease of use, that also enforces nonlinear wheel-ground contact models in an unprecedented DAE formulation. Furthermore, unlike ODE, our method is designed for efficiency specifically for WMRs. This is evident in our use of joint space dynamics algorithms, a limited collision engine, and inertia and bias force approximations as explained in Section 3.

Outside of the WMR application, there is relevant work on dynamic simulation. Some have published on the simulation of dynamic systems with *nonholonomic* constraints [18][25], of which wheel slip constraints are a type. Some have created or proposed modular simulators for space applications, such as SpaceDyn [24] and LRMAS [4]. Our method extends the spatial vector algorithms Featherstone originally developed for manipulator dynamics [8]. Orin applied these algorithms to a multilegged vehicle [19], but to our knowledge none have applied them to WMRs.

3 Simulation Mathematics

This section explains the mathematics of our simulation method, throughout which we use the following notational conventions:

- underline denotes a column vector of any dimension $\underline{u}$
- overset harpoon denotes a 3D Cartesian coordinate vector $\vec{u}$
- overset arrow denotes a 6D Plücker coordinate spatial vector $\vec{u}$
- ${}^{c}u_{a}^{b}$ indicates that the quantity u is of frame a with respect to frame b, expressed in the coordinates associated with frame c. u_{a}^{b} implies ${}^{b}u_{a}^{b}$.
- R_{a}^{b} denotes the rotation of frame a relative to frame b. Matrix multiplication is used to transform the coordinates that vectors are expressed in as follows: ${}^{b}\vec{u} = R_{a}^{b}({}^{a}\vec{u})$. Homogeneous and Plücker transforms (T,X) encode rotation and translation and use the same script notation.
- $[\vec{u}]_{\times}$ denotes a 3×3 skew symmetric matrix formed from the elements of $\vec{u}$. This is used to represent cross products by matrix multiplication: $\vec{a} \times \vec{b} = [\vec{a}]_{\times}\vec{b}$
- $\underline{u}(i)$ denotes the i^{th} element of the vector $\underline{u}$. $\underline{u}(1:n)$ denotes a subvector comprised of elements 1 through n. $A(i,j)$ denotes the element of matrix A at row i, column j. $A(i,*)$ denotes the i^{th} row, and $A(*,j)$ denotes the j^{th} column of A.

3.1 Kinematic Model and State Space

First, a kinematic model of the WMR is constructed as a tree of frames. The root is the body frame which has 6 degrees of freedom (DOF) with respect to the navigation/world frame. Additional frames for steering, suspension, etc. are attached via 1-DOF revolute or prismatic joints. All branches terminate with wheel frames,

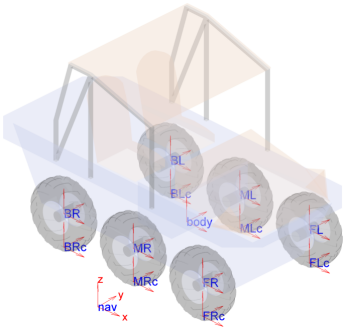


Fig. 1 LandTamer frames diagram

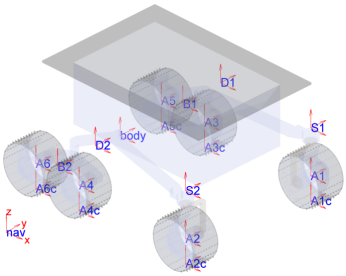


Fig. 2 Rocky 7 frames diagram

Table 1 LandTamer frames table. Made by PFM Manufacturing Inc.

i	Frame	Parent	Type	Act.	x	y	z	θ_x	θ_y	θ_z
1	body	nav								
2	FL	body	RY	Y	1	w	0	0	0	0
3	FR	body	RY	Y	1	-w	0	0	0	0
4	ML	body	RY	Y	0	w	0	0	0	0
5	MR	body	RY	Y	0	-w	0	0	0	0
6	BL	body	RY	Y	-l	w	0	0	0	0
7	BR	body	RY	Y	-l	-w	0	0	0	0

in inches: l=42, w=32.25, wheel radius=16.5, total mass = 3225 lbm

Table 2 Rocky 7 frames table [22]

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

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

which by convention are attached via revolute joints about their y-axes. Mass properties are also specified for each frame.

At each time step, a massless frame is also attached to each wheel in contact with the terrain. The origin of the contact frame is the point on the circumference of the wheel that most penetrates the terrain surface. This is computed by a limited collision engine that intersects wheel and surface geometries. Wheels can be represented as 3D circles (discretized into points) and surfaces can be bounded planes, elevation grids, triangular meshes, etc. The contact frame z-axis is aligned with the surface normal vector at the contact point. The x and y axes are aligned with the longitudinal and lateral slip directions.

Frame information is stored in an ordered list such that the index of any frame is greater than the index of its parent ($i > p(i)$, $i = 1$ is the body frame). Frame data for two example WMRs is provided in Tables 1 and 2. Joint types are revolute (R) or prismatic (P) about one of the axes (X,Y,Z). Act. means Actuated. The last six columns specify the pose of each frame with respect to its parent frame when joint displacement is zero.

We chose to use generalized (or reduced) coordinates for our method. The first elements of the state vector ($\underline{q}$) are the pose of the body frame with respect to the world frame ($\underline{\rho}$). Orientation ($\underline{o}$) may be expressed using either Euler angles or quaternions. The subsequent elements of the state vector are joint displacements ($\underline{\theta}$) in the same order as the frames list.

$$\underline{q} = \begin{bmatrix} \underline{\rho} \\ \underline{\theta} \end{bmatrix} \quad \underline{\rho} = \begin{bmatrix} \vec{t} \\ \underline{o} \end{bmatrix} \tag{1}$$

Open Dynamics Engine and Baraff [2] use a different state space which contains the 6-DOF pose of each frame. This necessitates numerous constraint equations for lower pairs; each 1-DOF revolute or prismatic joint requires a 5-DOF constraint. Big matrices must be inverted at every time step to solve for Lagrange multipliers, but computational cost can be mitigated somewhat by exploiting sparsity.

Given the frames list data and state vector one can compute the forward kinematics, or the homogeneous transform between each frame and its parent frame ($T_i^{p(i)}$) or the navigation frame (T_i^n). Homogeneous transforms are 4×4 matrices that encode rotation and translation:

$$T_a^b = \begin{bmatrix} R_a^b & \vec{t}_a^b \\ \underline{0}^T & 1 \end{bmatrix} \tag{2}$$

3.2 Spatial Vector Algebra Dynamics Algorithms

We express WMR dynamics using spatial vector algebra as published by Featherstone [8][9]. This is compatible with our prior work on a vector algebra formulation of WMR kinematics [15]. Spatial vectors are 6D and inhabit two vector spaces: motion and force. Spatial velocity and acceleration ($\vec{v}$,$\vec{a}$) are motion vectors; they contain 3D angular and linear components (on top and bottom respectively). Spatial force ($\vec{f}$) likewise contains 3D moment and linear force components.

$$\vec{v} = \begin{bmatrix} \vec{\omega} \\ \vec{v} \end{bmatrix} \quad \vec{a} = \begin{bmatrix} \vec{\alpha} \\ \vec{a} \end{bmatrix} \quad \vec{f} = \begin{bmatrix} \vec{\tau} \\ \vec{f} \end{bmatrix} \tag{3}$$

The unconstrained dynamic equation is:

$$M\underline{\ddot{q}}_s + \underline{c}(\underline{q}, \underline{\dot{q}}_s) = \underline{\tau} \tag{4}$$

$\underline{\dot{q}}_s$ is equivalent to the first time derivative of state ($\underline{\dot{q}}$), except that the time derivative of pose ($\underline{\dot{\rho}}$) is replaced with the spatial velocity of the body frame ($^b\vec{v}_b^n$). Likewise $\underline{\ddot{q}}_s$ contains the spatial acceleration. $\underline{\tau}$ is a vector of actuator torques/forces applied at the joints.

We use the Recursive Newton-Euler Algorithm (RNEA) to compute the joint space bias force $\underline{c}$, which includes the Coriolis and centripetal force terms, as shown in Algorithm 1. External wheel contact forces may be added directly on line 10 (as do [1][19][11]). Instead, we include these forces via constraints as explained in Section 3.3. Instead of adding gravitational force to each frame, we simply accelerate the base (line 2). n_f is the total number of frames, and n_w is the number of wheel frames. $X_{p(i)}^i$ is a 6×6 Plücker transform that converts *motion* spatial vectors from parent to child coordinates; its transpose converts *force* spatial vectors from child

to parent coordinates. I_i is the 6×6 spatial inertia of frame i (which encodes mass, center of mass, moment of inertia). The function $s(i)$ maps the joint type of frame i to a spatial vector index (RX=1, RY=2, RZ=3, PX=4, PY=5, PZ=6).

We use the Composite-Rigid-Body Algorithm (CRBA) to compute the joint space inertia, as shown in Algorithm 2. I_i^c denotes the composite inertia of the subtree rooted at frame i. Note that M is symmetric. Explanations of the original RNEA and CRBA are available in [9]. Our Algorithms 1 and 2 are modified to account for the special structure of WMR kinematics: a non-fixed base, only 1-DOF joints, and n_w contact frames at the list's end which are massless and fixed with respect to their parent wheel frames.

Algorithm 1 RNEA, joint space bias force

$\vec{v}_1 = \dot{\underline{q}}_s(1:6)$
$\vec{a}_1 = {}^b\vec{g}$
for $i = 1$ to n_f **do**
  if $i > 1$ **then**
    $\vec{h} = 0$
    if $i \leq (n_f - n_w)$ **then** $\vec{h}(s(i)) = \dot{\underline{q}}_s(i+5)$
    $\vec{v}_i = X^i_{p(i)}\vec{v}_{p(i)} + \vec{h}$
    $\vec{a}_i = X^i_{p(i)}\vec{a}_{p(i)} + \vec{v}_i \times \vec{h}$
  end if
  $\vec{f}_i = I_i\vec{a}_i + \vec{v}_i \times I_i\vec{v}_i (+\vec{f}_i^{ext})$
end for
for $i = n_f$ to 2 by -1 **do**
  if $i \leq (n_f - n_w)$ **then** $\underline{c}(i+5) = \vec{f}_i(s(i))$
  $\vec{f}_{p(i)} = \vec{f}_{p(i)} + (X^i_{p(i)})^T\vec{f}_i$
end for
$\underline{c}(1:6) = \vec{f}_1$

Algorithm 2 CRBA, joint space inertia

for $i = 1$ to $(n_f - n_w)$ **do** $I_i^c = I_i$
for $i = (n_f - n_w)$ to 2 by -1 **do**
  $I^c_{p(i)} = I^c_{p(i)} + (X^i_{p(i)})^T I_i^c (X^i_{p(i)})$
end for
$M = 0$
$M(1:6, 1:6) = I_1^c$
for $i = 2$ to $(n_f - n_w)$ **do**
  $\vec{f}^c = I_i^c(*, s(i))$
  $M(i+5, i+5) = \vec{f}^c(s(i))$
  $j = i$
  while $j > 1$ **do**
    $\vec{f}^c = (X^i_{p(i)})^T\vec{f}^c$
    $j = p(j)$
    if $j = 1$ **then**
      $M(1:6, i+5) = \vec{f}^c$
      $M(i+5, 1:6) = M(i+5, 1:6)^T$
    else
      $M(j+5, i+5) = \vec{f}^c(s(j))$
      $M(i+5, j+5) = M(j+5, i+5)$
    end if
  end while
end for

3.3 Wheel-ground Contact Constraints

Each wheel-ground contact frame has three constraints: one holonomic surface contact constraint which restricts motion along its z-axis, and two nonholonomic slip velocity constraints which restrict motion along its x and y axes. As in [25], holonomic constraints are converted to velocity constraints by differentiation. For a sin-

gle wheel, the constraint equations are of the form:

$$A\dot{\underline{q}}_s = \vec{v}_c \tag{5}$$

Algorithm 3 Wheel constraint matrix

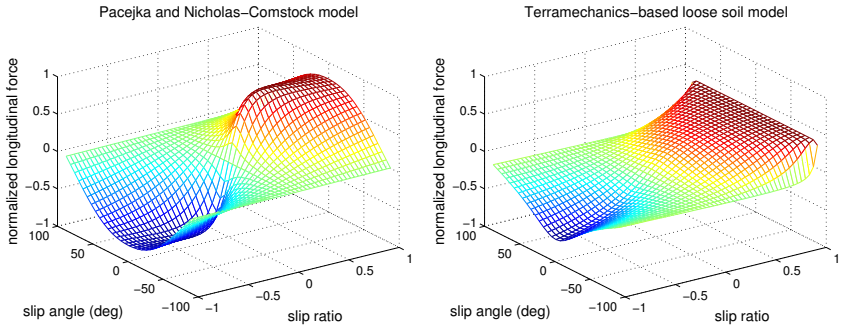
1: $c =$ contact frame index
2: $i = p(c)$
3: **while** $i > 1$ **do**
4: **if** $s(i) \in \{1,2,3\}$ **then**
5: $A(*, i+5) = R_i^n(*, s(i)) \times (\vec{t}_c^{\,n} - \vec{t}_i^{\,n})$
6: **else if** $s(i) \in \{4,5,6\}$ **then**
7: $A(*, i+5) = R_i^n(*, s(i)-3)$
8: **end if**
9: $i = p(i)$
10: **end while**
11: $A(*, 1:3) = [\vec{t}_c^{\,n} - \vec{t}_1^{\,n}]_\times^T R_b^n$
12: $A(*, 4:6) = R_b^n$
13: $A = R_n^c A$

The matrix A is computed by Algorithm 3, which works backwards along the kinematic chain from contact to body frame. On line 11, the identity $\vec{\omega} \times \vec{t} = -\vec{t} \times \vec{\omega} = [\vec{t}]_\times^T \vec{\omega}$ is used, as $\dot{\underline{q}}_s$ contains the angular velocity of the body frame. The right-hand side $\vec{v}_c$ is short for ${}^c\vec{v}_c^{\,n}$: the velocity of the contact frame with respect to the ground expressed in contact coordinates. This is not constant, but is solved for by optimization as explained in Section 3.4.

We express all wheel-ground contact models as functions in a common format:

$$\vec{f}_c = f(\vec{v}_c, R\omega, \Delta z) \tag{6}$$

The forces exerted by the ground on the wheel ($\vec{f}_c$) are dependent on $\vec{v}_c$, the product of wheel radius and angular rate ($R\omega$), and the displacement between the contact point and terrain surface (Δz) due to sinkage or compression. Plots of longitudinal force vs. slip ratio and angle for two example models are shown in Figure 3.



(a) (b)

Fig. 3 Normalized longitudinal force (f_x/f_z) vs. slip ratio and angle for two wheel-ground contact models. (a) and (b) use equations and parameters in [3] and [11] respectively. These plots are for fixed Δz. Though not shown, these models also determine lateral and normal force.

Constraints for all wheels are stacked into one matrix equation with $3n_w$ rows. Hereafter let A denote the stacked matrix and $\underline{v}_c$ the stacked vector for all wheel constraints. Additional constraints may be appended to account for mechanical restrictions on joint displacements (such as roll/pitch averaging) or to enforce desired speeds for actuated joints.

3.4 Force-balance Optimization

This section explains perhaps this paper's most important theoretical contribution, the enforcement of *nonlinear* wheel-ground contact models in a DAE formulation. The dynamics equation (4) is modified to include constraints as follows:

$$\begin{bmatrix} M & A^T \\ A & C \end{bmatrix} \begin{bmatrix} \underline{\ddot{q}}_s \\ \underline{\lambda} \end{bmatrix} = \begin{bmatrix} \underline{\tau} - \underline{c} \\ \underline{b} \end{bmatrix} \tag{7}$$

where C is a matrix of zeros except for potentially non-zero "constraint force mixing" values on the diagonal for appended holonomic constraints on joint displacements. These are used just as in Open Dynamics Engine to introduce compliance. (7) can be rearranged to solve for the vector of Lagrange multipliers ($\underline{\lambda}$) which represent constraint forces:

$$\underline{\lambda} = [AM^{-1}A^T + C]^{-1}(\underline{b} - AM^{-1}(\underline{\tau} - \underline{c})) \tag{8}$$

Note that the constraints on state velocity ($\underline{\dot{q}}_s$) have been converted to constraints on state acceleration ($\underline{\ddot{q}}_s$) as follows:

$$A\underline{\dot{q}}_s[i+1] = \underline{v}_c[i+1] \tag{9}$$

$$A(\underline{\dot{q}}_s[i] + \underline{\ddot{q}}_s \Delta t) = \underline{v}_c[i+1] \tag{10}$$

$$A\underline{\ddot{q}}_s = (\underline{v}_c[i+1] - A\underline{\dot{q}}_s[i])/\Delta t \tag{11}$$

$$A\underline{\ddot{q}}_s = (\underline{v}_c[i+1] - \underline{v}_c[i])/\Delta t = \underline{b} \tag{12}$$

$[i]$ and $[i+1]$ denote the current and next time step. $\underline{v}_c[i]$ is already computed in Algorithm 1, whereas $\underline{v}_c[i+1]$ must be computed by optimization:

$$\underset{\underline{v}_c[i+1]}{\arg\min} \, \|\underline{\lambda}(i | i \in W) - \underline{f}_c\| \tag{13}$$

In short, contact point velocities are chosen such that constraint forces computed by the dynamics equation (8) match those computed by the wheel-ground contact model (6). $\underline{f}_c$ denotes the stacked vector of contact model forces for all wheels. W denotes the set of wheel constraint indices (excludes appended constraints).

This optimization can be performed efficiently using Newton's method. Let $\underline{x}$ denote the argument $\underline{v}_c[i+1]$ and $f(\underline{x})$ the objective function; then our guess for $\underline{x}$

is updated as follows:

$$\underline{x}_{n+1} = \underline{x}_n - \gamma[Hf(\underline{x}_n)]^{-1}\nabla f(\underline{x}_n), \quad 0 \leq \gamma \leq 1 \tag{14}$$

Computing the Hessian ($Hf(\underline{x}_n)$) and gradient ($\nabla f(\underline{x}_n)$) requires the Jacobian of the wheel-ground contact model output $\underline{f}_c$ with respect to its inputs. The step size γ is chosen such that the strong Wolfe conditions are satisfied, using a line search algorithm like Algorithm 3.2 in [21].

WMRs have six DOF for motion of the body frame, plus one DOF for each revolute/prismatic joint. They have three constraints for each wheel in contact, plus any appended constraints. Many WMR configurations are overconstrained which results in a rank-deficient A matrix. To address this we choose and enforce a linearly independent subset of the constraints. A well-conditioned subset can be chosen by QR decomposition of A^T. The subset contains all appended constraints, but only some of the wheel constraints. $\underline{f}_c$ in the objective function (13) is replaced with $P\underline{f}_c$ where P projects the full vector of contact forces onto the subset space.

Once $\underline{\ddot{q}}_s$ is solved for, state velocity and state can be updated using symplectic Euler integration as follows:

$$\underline{\dot{q}}_s[i+1] = \underline{\dot{q}}_s[i] + \underline{\ddot{q}}_s \Delta t \tag{15}$$

$$\underline{\dot{q}}[i+1] \leftarrow \underline{\dot{q}}_s[i+1] \tag{16}$$

$$\underline{q}[i+1] = \underline{q}[i] + \underline{\dot{q}}[i+1]\Delta t \tag{17}$$

Note that (16) requires the conversion of angular velocity to either Euler angle or quaternion rates. Higher-order integration methods such as Runge-Kutta are also possible. Unlike the "stiff" ordinary differential equation method, our DAE method is stable even for large time steps.

3.5 Recommendations for Computational Speed-up

No matter how fast your processor, a faster simulator can improve planning performance by enabling more candidate trajectories to be evaluated. One can improve computation time in several ways without compromising simulation accuracy. First, in the optimization initialize contact point velocities ($\underline{v}_c[i+1]$) to values at the previous time step ($\underline{v}_c[i]$). WMRs frequently execute steady-state maneuvers during which these change little. Specify a cost threshold below which optimization via Newton's method is not required. Next, precompute lookup tables for the wheel-ground contact model and its Jacobian. This is particularly beneficial for the terramechanics-based models in [11][14], which require the costly integration of stress distributions along the wheel surface.

One can further speed up computation for reasonable compromises in accuracy. First, changes in the joint space inertia matrix M may have negligible impact on WMR motion within the predictive horizon. If so, only compute M and M^{-1} for the

first time step and reuse these values on subsequent steps. Additionally, the effect of Coriolis and centripetal forces on internal articulations may be negligible. If so, the joint space bias force $\underline{c}$ can be approximated by a vector of zeros except for:

$$\underline{c}(1:6) = I_1^c({}^b\vec{g}) + \vec{v}_1 \times I_1^c \vec{v}_1 \tag{18}$$

This considers the WMR to be a single rigid body (with all joints locked). I_1^c is the composite inertia of the WMR rooted at the body frame, which like M can be computed only once.

4 Results

In this section we evaluate our simulation method in three tests. For the first test, we simulate the LandTamer vehicle (Figure 1) with a Pacejka wheel-ground contact model (Figure 3(a)) driving over a 20° ramp then turning to the left and right at 2 m/s (Figure 4). Figure 5 shows slip ratios and angles for two simulations of this trajectory: one using our method of handling contact forces via constraints, the other adding them directly. Using constraints eliminates jitter and reduces computation time from 21.40s to 2.03s (implemented in MATLAB, using a 2.83 GHz processor). Both simulations benefit from our method's reduced state space and use of the RNEA/CRBA; this speed-up will be quantified in future work. Both simulations use constraints to control wheel velocities, as PID torque controllers can make the dynamics very stiff.

An adaptive integrator (MATLAB's ode45) is required for the direct addition method. Figure 6(a) shows that, to prevent jitter, the integrator takes very small steps relative to the .04s steps taken by our method. If .04s steps were taken by the direct addition method, jitter would become severe instability. Figure 6(b) shows the number of Newton's method iterations required for force-balance optimization in our method (Section 3.4); zero iterations are required during steady-state periods.

We also validated our dynamic model of the LandTamer in a physical experiment (the second test). We drove the LandTamer in various arcs (at up to 2.5 m/s and 0.5 rad/s) in a parking lot at the Taylor test site near Pittsburgh, PA. We tuned parameters of the dynamic model (including tire model parameters and the center of mass) to fit one portion of the dataset, then evaluated on the remainder (Figure 7). Table 3 compares the mean position and absolute yaw error for our tuned dynamic model

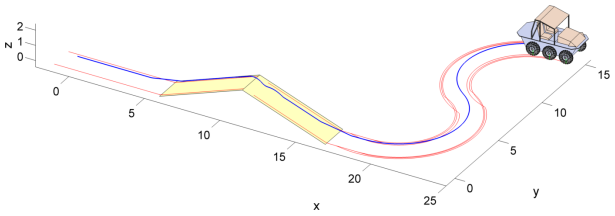
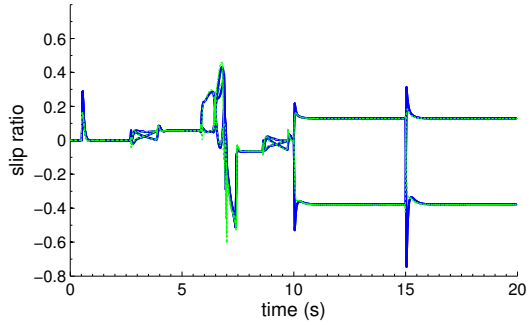


Fig. 4 LandTamer animation screenshot. The path of the body frame is traced in blue; the paths of wheel-ground contact points are traced in red.



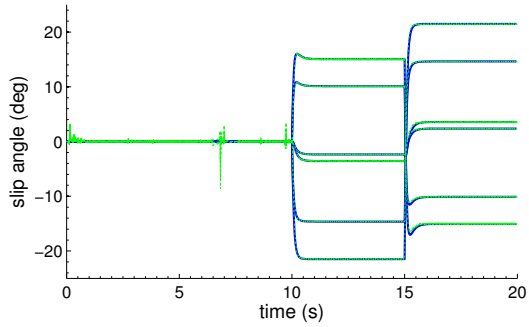
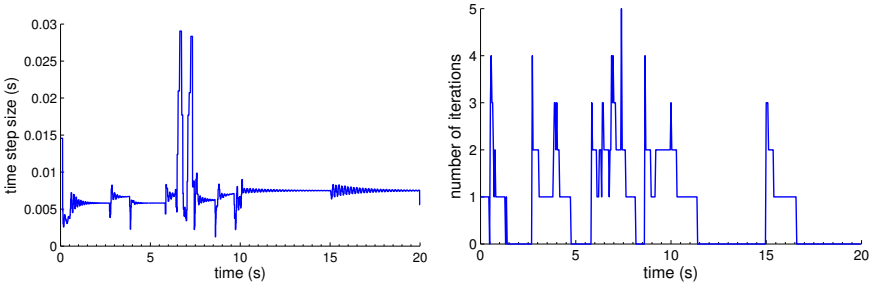


Fig. 5 Slip ratios and angles for all wheels for two simulations of the LandTamer executing the trajectory shown in Figure 4. Our method of handling contact forces via constraints (solid lines) reduces jitter compared to the more common method of adding contact forces directly (dashed lines). As expected, the slip ratios are positive while ascending the ramp (4-6s) and negative while descending (7-9s) due to gravity. The slip angles indicate lateral slip away from the center of curvature when turning (10-20s).



(a) (b)

Fig. 6 (a) Integration time step size for the direct addition method (compare to .04s for our method). (b) Number of Newton's method iterations required for our method.

and a kinematic model that minimizes slip velocity at the six wheels, i.e. $||\underline{v}_c||$. For reasonable planning horizons of 5-20m our predicted position error is 1-2% of distance traveled. Both position and yaw prediction errors using our dynamic model are 88-93% less than using the kinematic model.

For the third test, we simulate the Rocky 7 rover (Figure 2) with a terramechanics-based wheel-ground contact model (Figure 3(b)). The rover traverses uneven, random terrain while making wide turns for 10 seconds at 0.5 m/s (Figure 8). Figure

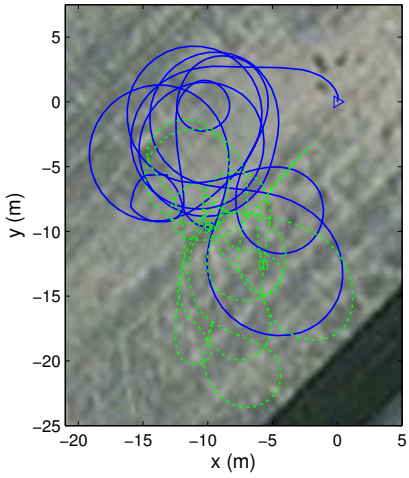


Fig. 7 Path of the LandTamer at the Taylor test site parking lot, overlaid on a satellite image (dashed-training, solid-evaluation)

Table 3 Model prediction error for the Taylor dataset

	dynamic model		kinematic model	
horizon	pos.	|yaw|	pos.	|yaw|
5m	0.0669	0.0169	0.5442	0.2006
10m	0.1470	0.0292	1.8076	0.3818
20m	0.3839	0.0515	5.1336	0.6796

Position (m) and absolute yaw (rad) error values are averages over at least 50 trials. Data for these trials were obtained by subdividing the 273m evaluation path into equally spaced segments; 10 and 20m segments overlapped.

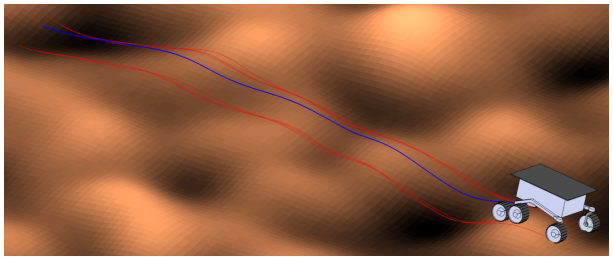
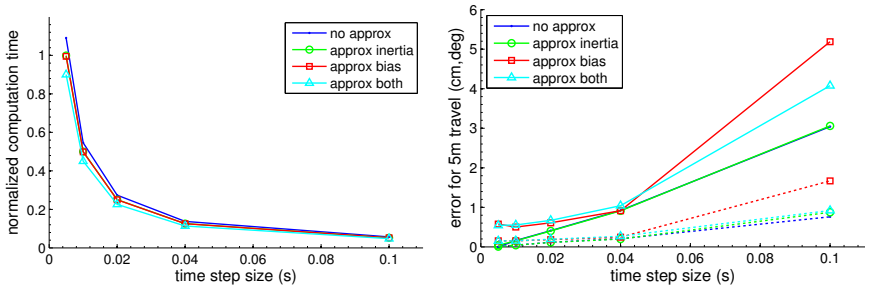


Fig. 8 Rocky 7 animation screenshot.

9(a) shows how computation time decreases exponentially with larger time step size. Computation times are normalized by dividing by simulation time; values less than one indicate faster than real time. Each of the approximations suggested in Section 3.5 reduces computation time by approximately 10%. In Figure 9(b), error is the difference in predicted terminal pose with respect to the prediction using no approximations and the minimum (.005s) time step size, for 5m of travel. Error increases linearly (not exponentially) with time step size, and only modestly with approximation. Modest errors may be an acceptable tradeoff for significant speed-up in some planning applications.

5 Conclusions and Future Work

We presented a modular method for the simulation of wheeled mobile robots. In contrast to existing resources such as Open Dynamics Engine, our method uses spatial vector algebra dynamics algorithms which are particularly efficient for WMRs. More importantly, in contrast to ODE, our method can accommodate complex, nonlinear wheel-ground contact models. Our enforcement of these models via con-



(a) (b)

Fig. 9 Computation time and error vs. time step size for our dynamics method, with/without approximation of the joint space inertia and bias force. Averaged over 30 trials. In (b) solid lines are for 2D position error, dashed lines are for absolute yaw error.

straints in a DAE formulation was demonstrated to be more stable and efficient than the common method of directly adding contact forces in an ordinary differential equation formulation (Section 4, first test).

More physical experiments will be presented in future work on the calibration of 3D WMR motion models. We plan to convert our MATLAB implementation of the WMR simulator to C++; only then can we fairly compare computational speed with alternatives (like ODE, CarSim, etc.). We also plan to make this software publicly available. This will enable existing and future research on wheel-ground contact models to be readily applied to the prediction of full vehicle mobility. The simulator should be fast and accurate enough for improved model-predictive planning in many challenging applications.

Acknowledgements This research was made with U.S. Government support by the Army Research Laboratory (W911NF-10-2-0016) and by the National Science Foundation Graduate Research Fellowship (0946825). Our colleague Forrest Rogers-Marcovitz collected data at the Taylor test site.

References

1. Balakrishna, R., Ghosal, A.: Modeling of Slip for Wheeled Mobile Robots. IEEE Transactions on Robotics and Automation **11**(1) (1995)
2. Baraff, D.: Linear-time Dynamics Using Lagrange Multipliers. In: Proc. SIGGRAPH, pp. 137–146 (1996)
3. Brach, R.M., Brach, R.M.: Tire Models for Vehicle Dynamic Simulation and Accident Reconstruction. SAE Technical Paper (2009)
4. Ding, L., Gao, H., Deng, Z., Song, P., Liu, R.: Design of Comprehensive High-fidelity/High-speed Virtual Simulation System for Lunar Rover. In: Proc. IEEE Conference on Robotics, Automation and Mechatronics, pp. 1118–1123 (2008)
5. Ding, L., Yoshida, K., Nagatani, K., Gao, H., Deng, Z.: Parameter Identification for Planetary Soil Based on a Decoupled Analytical Wheel-Soil Interaction Terramechanics Model.

In: Proc. IEEE International Conference on Intelligent Robots and Systems, pp. 4122–4127 (2009)
6. Drumwright, E., Hsu, J., Koenig, N., Shell, D.: Extending Open Dynamics Engine for Robotics Simulation. In: Proc. Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), pp. 38–50 (2010)
7. Eathakota, V., Aditya, G., Krishna, M.: Quasi-static motion planning on uneven terrain for a wheeled mobile robot. In: Proc. IEEE International Conference on Intelligent Robots and Systems, pp. 4314–4320 (2011)
8. Featherstone, R.: Robot dynamics algorithms. Kluwer Academic Publishers, Boston/Dordrecht/Lancaster (1987)
9. Featherstone, R., Orin, D.: Robot dynamics: equations and algorithms. In: Proc. IEEE International Conference on Robotics and Automation, pp. 826–834 (2000)
10. Howard, T.: Adaptive model-predictive motion planning for navigation in complex environments. Carnegie Mellon University, Robotics Institute Tech. Report, CMU-RI-TR-09-32 (2009)
11. Ishigami, G., Miwa, A., Nagatani, K., Yoshida, K.: Terramechanics-Based Model for Steering Maneuver of Planetary Exploration Rovers on Loose Soil. Journal of Field Robotics **24**(3), 233–250 (2007)
12. Ishigami, G., Nagatani, K., Yoshida, K.: Path Planning and Evaluation for Planetary Rovers Based on Dynamic Mobility Index. In: Proc. IEEE International Conference on Intelligent Robots and Systems, pp. 601–606 (2011)
13. Jain, A., Guineau, J., Lim, C., Lincoln, W., Pomerantz, M., Sohl, G., Steele, R.: ROAMS: Planetary Surface Rover Simulation Environment. In: Proc. International Symposium on Artificial Intelligence, Robotics and Automation in Space (2003)
14. Jia, Z., Smith, W., Peng, H.: Fast Computation of Wheel-Soil Interactions for Safe and Efficient Operation of Mobile Robots. In: Proc. IEEE International Conference on Intelligent Robots and Systems, pp. 3004–3010 (2011)
15. Kelly, A., Seegmiller, N.: A Vector Algebra Formulation of Mobile Robot Velocity Kinematics. In: Proc. Field and Service Robotics (2012)
16. Knepper, R.: On the fundamental relationships among path planning alternatives. Carnegie Mellon University, Robotics Institute Tech. Report, CMU-RI-TR-11-19 (2009)
17. Lamon, P., Siegwart, R.: 3D Position Tracking in Challenging Terrain. International Journal of Robotics Research **26**(2), 167–186 (2007)
18. Luca, A.D., Oriolo, G.: Chapter 7: Modeling and Control of Nonholonomic Mechanical Systems. In: Kinematics and Dynamics of Multi-Body Systems, pp. 277–342. Springer Verlag (1995)
19. McMillan, S., Orin, D.E.: Forward Dynamics of Multilegged Vehicles Using the Composite Rigid Body Method. In: Proc. IEEE International Conference on Robotics and Automation, May, pp. 464–470 (1998)
20. Muir, P.: Modeling and control of wheeled mobile robots. Carnegie Mellon University, Robotics Institute Tech. Report, CMU-RI-TR-88-20 (2009)
21. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
22. Tarokh, M., McDermott, G.: Kinematics modeling and analyses of articulated rovers. IEEE Transactions on Robotics **21**(4), 539–553 (2005)
23. Tian, Y., Sidek, N., Sarkar, N.: Modeling and Control of a Nonholonomic Wheeled Mobile Robot with Wheel Slip Dynamics. In: Proc. IEEE Symposium on Computational Intelligence in Control and Automation (2009)
24. Yoshida, K.: The SpaceDyn: a MATLAB toolbox for space and mobile robots. In: Proc. IEEE International Conference on Intelligent Robots and Systems, pp. 1633–1638 (1999)
25. Yun, X., Sarkar, N.: Unified Formulation of Robotic Systems with Holonomic and Nonholonomic Constraints. IEEE Transactions on Robotics **14**(4), 640–650 (1998)