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 in which 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.

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 inertias 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 in the specified format.

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.

Our method extends 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 (often 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 often be "stiff" requiring a solver that takes very small or adaptive time steps [1]. In contrast, we enforce wheel-ground contact models in a differential algebraic equation (DAE) formulation using Lagrange multipliers. This requires the resolution of nonlinearity and overconstraint issues, but ultimately proves to be more stable and efficient. 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 available simulator resources are unsuitable.

Some commercial resources exist for vehicle simulation such as Adams/Car and CarSim. These use complex, granular models that include many subsystems and components; 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 sometimes 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 formula-

tion. Furthermore, unlike ODE, our method is designed to be efficient 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 their previous application to WMRs is limited.

3 Simulation Mathematics

This section explains the mathematics of our simulation method, but first we provide notes on notation 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 measurement u is of object a with respect to b, expressed in the coordinates associated with frame c
- R^{a}_{b} denotes a rotation matrix that changes coordinates as follows: ${}^{a}\vec{u} = R^{a}_{b}({}^{b}\vec{u})$. The same script notation applies to homogeneous and Plücker transforms (T,X)
- $[\vec{u}]_{\times}$ 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, $A(*,j)$ denotes the j^{th} column.

Notation change. In prior work $\vec{u}$ means not expressed in the coordinates of any particular reference frame

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 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, 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

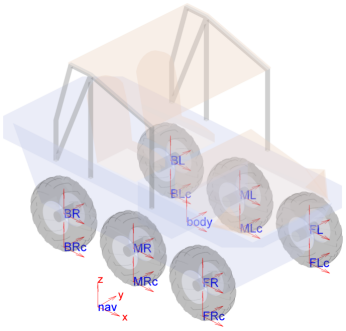


Fig. 1 LandTamer frames diagram

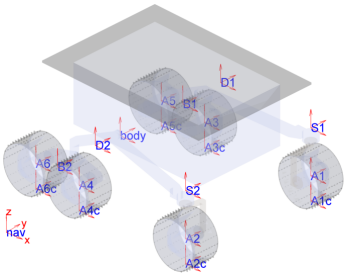


Fig. 2 Rocky 7 frames diagram

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

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 contact frame x-axis is aligned with the cross product of the normal vector and the y-axis of the parent wheel frame. Finally the contact frame y-axis is aligned with the cross product of the z and x axes.

Frame information is stored in an ordered list such that the index of any frame is greater than the index of its parent ($p(i) < i$, $i = 1$ 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 cost can be mitigated somewhat by exploiting sparsity.

Algorithm 1 shows how, given the frames list data and state vector, one can compute homogeneous transforms between each frame and its parent frame or the navigation frame. These are required in subsequent algorithms. There are n_f frames in the list (including contact frames). The function $a(i)$ returns a number based on joint type (RX=1, RY=2, RZ=3, PX=4, PY=5, PZ=6). *Rot* (line 7) returns a rotation matrix given an axis number (1-3) and angle.

Algorithm 1 Calculation of homogeneous transforms

1: $\{{}^n\vec{t}^b_n, \underline{o}, \underline{\theta}\} \leftarrow \underline{q}$
2: $R^n_b \leftarrow \underline{o}$
3: $T^n_b = \begin{bmatrix} R^n_b & {}^n\vec{t}^b_n \\ \underline{0}^T & 1 \end{bmatrix}$
4: **for** $i = 2$ to n_f **do**
5: initialize $T_i^{p(i)}$ as if $\underline{\theta}(i-1) = 0$
6: **if** $a(i) \in \{1,2,3\}$ **then**
7: $T_i^{p(i)}(1:3,1:3) = T_i^{p(i)}(1:3,1:3)Rot(a(i), \underline{\theta}(i-1))$
8: **else if** $a(i) \in \{4,5,6\}$ **then**
9: $T_i^{p(i)}(1:3,4) = T_i^{p(i)}(1:3,4) + T_i^{p(i)}(a(i)-3,1:3)^T \underline{\theta}(i-1)$
10: **end if**
11: **end for**
12: **for** $i = 2$ to n_f **do** $T^n_i = T^n_{p(i)} T_i^{p(i)}$

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 are motion vectors; they contain 3D angular and linear components. Spatial force likewise contains 3D moment and linear force components.

$$\vec{v} = \begin{bmatrix} \vec{\omega} \\ \vec{v} \end{bmatrix} \quad \vec{a} = \begin{bmatrix} \vec{\alpha} \\ \vec{a} \end{bmatrix} \quad \vec{f} = \begin{bmatrix} \vec{m} \\ \vec{f} \end{bmatrix} \tag{2}$$

The unconstrained dynamic equation is:

$$M\underline{\ddot{q}}^s + \underline{c}(\underline{q}, \underline{\dot{q}}^s) = \underline{\tau} \tag{3}$$

$\underline{\dot{q}}^s$ is equivalent to the first time derivative of state ($\underline{\dot{q}}$), except that the time derivative of pose ($\dot{\rho}$) is replaced with the spatial velocity of the body frame ${}^b\vec{v}_n^b$. 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 acceleration-independent Coriolis and centripetal force terms, as shown in Algorithm 2. When adding wheel contact forces directly (as do [1][19][11]), they must be added to $\vec{f}^i$ for all contact frames *before* the backwards traversal (line 11). Instead, we include these forces via constraints as explained in Section 3.3. Instead of adding the force of gravity to each frame, we simply accelerate the base (line 2). n_w is the number of wheel (and contact) frames. $X^i_{p(i)}$ 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).

Algorithm 2 Calculation of joint space bias force using RNEA

1: $\vec{v}^1 = \underline{\dot{q}}^s(1:6)$
2: $\vec{a}^1 = {}^b\vec{g}$
3: **for** $i = 1$ to n_f **do**
4: **if** $i > 1$ **then**
5: $\vec{h} = 0$
6: **if** $i \leq n_f - n_w$ **then** $\vec{h}(a(i)) = \underline{\dot{q}}^s(i+5)$
7: $\vec{v}^i = X^i_{p(i)}\vec{v}^{p(i)} + \vec{h}$
8: $\vec{a}^i = X^i_{p(i)}\vec{a}^{p(i)} + \vec{v}^i \times \vec{h}$
9: **end if**
10: $\vec{f}^i = I_i\vec{a}^i + \vec{v}^i \times I_i\vec{v}^i$
11: **end for**
12: **for** $i = n_f$ to 2 by -1 **do**
13: **if** $i \leq n_f - n_w$ **then** $\underline{c}(i+5) = \vec{f}^i(a(i))$
14: $\vec{f}^{p(i)} = \vec{f}^{p(i)} + (X^i_{p(i)})^T\vec{f}^i$
15: **end for**
16: $\underline{c}(1:6) = \vec{f}^1$

We use the Composite-Rigid-Body Algorithm (CRBA) to compute the joint space inertia, as shown in Algorithm 3. I^c_i denotes the composite inertia of the subtree rooted at frame i. Note that M is symmetric.

Algorithms 2 and 3 match those in [9], except that we account for the special structure of WMR kinematics: a non-fixed base, 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 3 Calculation of joint space inertia using CRBA

1: $I^c = I$
2: **for** $i = n_f - n_w$ to 2 by -1 **do** $I^c_{p(i)} = I^c_{p(i)} + (X^i_{p(i)})^T I^c_i (X^i_{p(i)})$
3: $M = 0$
4: $M(1:6,1:6) = I^c_1$
5: **for** $i = 2$ to $n_f - n_w$ **do**
6: $\quad \vec{f}^c = I^c_i(*, a(i))$
7: $\quad M(i+5, i+5) = \vec{f}^c_i(a(i))$
8: $\quad j = i$
9: $\quad$ **while** $j > 1$ **do**
10: $\quad\quad \vec{f}^c = (X^i_{p(i)})^T \vec{f}^c$
11: $\quad\quad j = p(j)$
12: $\quad\quad$ **if** $j = 1$ **then**
13: $\quad\quad\quad M(1:6, i+5) = \vec{f}^c, \quad M(i+5, 1:6) = M(i+5, 1:6)^T$
14: $\quad\quad$ **else**
15: $\quad\quad\quad M(j+5, i+5) = \vec{f}^c(a(j)), \quad M(i+5, j+5) = M(j+5, i+5)$
16: $\quad\quad$ **end if**
17: $\quad$ **end while**
18: **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 single wheel, the constraint equations are of the form:

$$A\dot{\underline{q}}^s = \vec{v}^c \tag{4}$$

The matrix A is computed according to Algorithm 4. Both revolute and prismatic joints are supported (lines 4, 6). On line 11, the identity $\vec{\omega} \times \vec{r} = -\vec{r} \times \vec{\omega} = [\vec{r}]^T_\times \vec{\omega}$ 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.

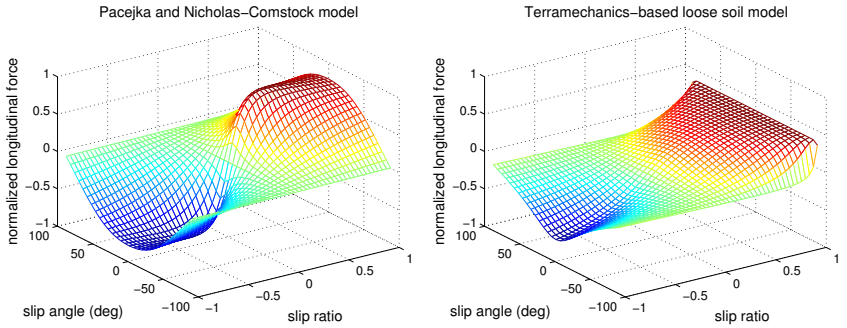
We express all wheel-ground contact models as functions in a common format:

$$\vec{f}_c = f(\vec{v}_c, R\omega, \Delta z) \tag{5}$$

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.

Algorithm 4 Calculation of wheel constraint matrix for a single wheel

c = contact frame index
$i = p(c)$
while $i > 1$ **do**
  if $a(i) \in \{1,2,3\}$ **then**
    $A(*, i+5) = R_i^n(*, a(i)) \times ({}^n\vec{t}_n^c - {}^n\vec{t}_n^i)$
  else if $a(i) \in \{4,5,6\}$ **then**
    $A(*, i+5) = R_i^n(*, a(i)-3)$
  end if
  $i = p(i)$
end while
$A(*, 1:3) = [{}^n\vec{t}_n^c - {}^n\vec{t}_n^1]_\times^T R_b^n$
$A(*, 4:6) = R_b^n$
$A = R_n^c A$



(a) (b)

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

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 physical restrictions on joint displacements (such as roll/pitch averaging) or to enforce desired speeds for actuated joints.

3.4 Force-balance Optimization

The dynamics equation (3) is modified to include constraints as follows:

$$\begin{bmatrix} M & A^T \\ A & C \end{bmatrix} \begin{bmatrix} \ddot{\underline{q}}^s \\ \underline{\lambda} \end{bmatrix} = \begin{bmatrix} \underline{\tau} - \underline{c} \\ \underline{b} \end{bmatrix} \tag{6}$$

where C is a matrix of zeros except for potentially non-zero "constraint force mixing" values on the diagonal for the appended holonomic constraints on joint displacements. These are used just as in Open Dynamics Engine to introduce compliance. (6) can be rearranged to solve for the vector of Lagrange multipliers ($\underline{\lambda}$) which represent constraint forces:

$$\underline{\lambda} = [AM^{-1}A^T + C]^{-1}(\underline{b} - AM^{-1}(\underline{\tau} - \underline{c})) \tag{7}$$

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:

$$A\underline{\dot{q}}^s[i+1] = \underline{v}^c[i+1] \tag{8}$$

$$A(\underline{\dot{q}}^s[i] + \underline{\ddot{q}}^s \Delta t) = \underline{v}^c[i+1] \tag{9}$$

$$A\underline{\ddot{q}}^s = (\underline{v}^c[i+1] - A\underline{\dot{q}}^s[i])/\Delta t \tag{10}$$

$$A\underline{\ddot{q}}^s = (\underline{v}^c[i+1] - \underline{v}^c[i])/\Delta t = \underline{b} \tag{11}$$

$[i]$ and $[i+1]$ denote the current and next time step. $\underline{v}^c[i]$ is already computed in Algorithm 2, whereas $\underline{v}^c[i+1]$ must be computed by optimization:

$$\underset{\underline{v}^c[i+1]}{\arg\min} \|\underline{\lambda}(i : i \in W) - \underline{f}^c\| \tag{12}$$

In short, contact point velocities are chosen such that constraint forces computed by the dynamics equation (6) match those computed by the wheel-ground contact model (5). $\underline{f}^c$ denotes the stacked vector of contact model forces for all wheels. W denotes the set of *wheel* constraint indices (does not include 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:

$$\underline{x}_{n+1} = \underline{x}_n - \gamma[Hf(\underline{x}_n)]^{-1}\nabla f(\underline{x}_n), \quad 0 \leq \gamma \leq 1 \tag{13}$$

Computing the Hessian ($Hf(\underline{x}_n)$) and gradient ($\nabla f(\underline{x}_n)$) requires the Jacobian of the wheel-ground contact model output 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 (12) is replaced with $P\underline{f}^c$ where P projects the full vector of contact forces onto the linearly independent subspace.

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

$$\underline{\dot{q}}^s[i+1] = \underline{\dot{q}}^s[i] + \underline{\ddot{q}}^s \Delta t \tag{14}$$

$$\underline{\dot{q}}[i+1] \leftarrow \underline{\dot{q}}^s[i+1] \tag{15}$$

$$\underline{q}[i+1] = \underline{q}[i] + \underline{\dot{q}}[i+1]\Delta t \tag{16}$$

Note that (15) 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 Speedup

One can improve computation time in several ways without compromising simulation accuracy. First, in the optimization initialize the guess of contact velocities ($\underline{v}^c[i+1]$) to those 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 speedup 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:

$$\underline{c}(1:6) = I_1^c({}^b\vec{g}) + \vec{v}^1 \times I_1^c \vec{v}^1 \tag{17}$$

This is like considering 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 validate our simulation method in two tests. In the first, we simulate the LandTamer vehicle (Figure 1) with a Pacejka wheel-ground contact model (Figure 3(a)). The LandTamer drives over a 20° ramp, then turns to the left and right. The slip ratios and angles for all wheels are shown in Figure 4. Our method

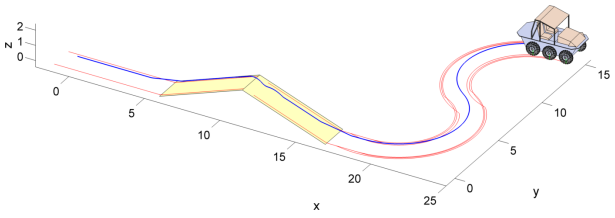


Fig. 4 LandTamer animation screenshot. The path of the body frame is traced in blue; the paths of wheel-ground contact points are traced in red.

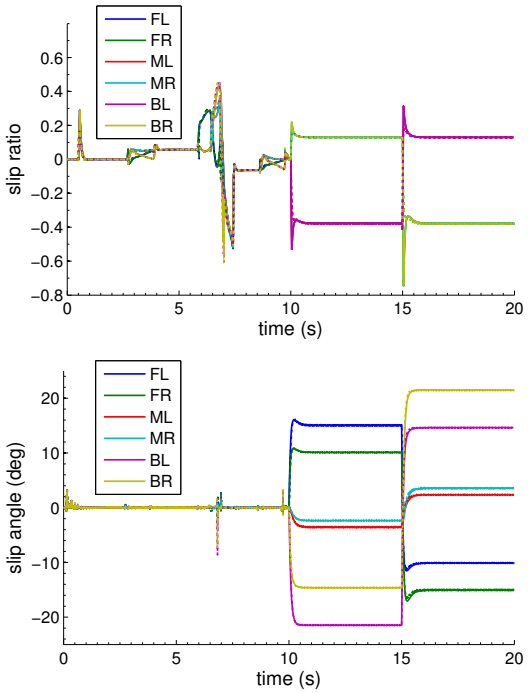
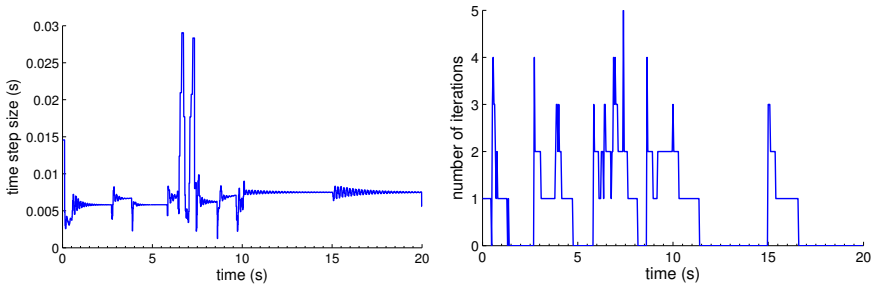


Fig. 5 Slip ratios and angles for the LandTamer simulation. Our method (solid lines) produces nearly identical output to the direct addition method (dashed lines), but without the jitter. As expected, the slip ratio is positive while ascending the ramp (4-6s) and negative while descending (7-9s) due to gravity. The slip angle indicates lateral slip away from the center of curvature when turning (10-20s).

produced identical output to the common method of adding contact forces directly, but at less cost. Our method reduced computation from 21.40 to 2.03 seconds (with a 2.83 GHz Intel processor). Of that time, the dynamics algorithms used only 0.98s (collision checking used the rest). Constraints were used in both methods to control wheel velocities, as PID control of wheel torques can make the dynamics very stiff.

An adaptive integrator (MATLAB's ode45) was required for the direct addition method. Figure 6(a) shows that, to prevent jitter, the integrator took very small steps relative to the .04s steps taken by our method. Figure 6(b) shows the number of Newton's method iterations required for our method. Notice that during steady-state periods, zero iterations are required.

In the second test, we simulate the Rocky 7 rover (Figure 2) with a terramechanics-based wheel-ground contact model (Figure 3(b)). The rover traverses uneven, ran-



(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.

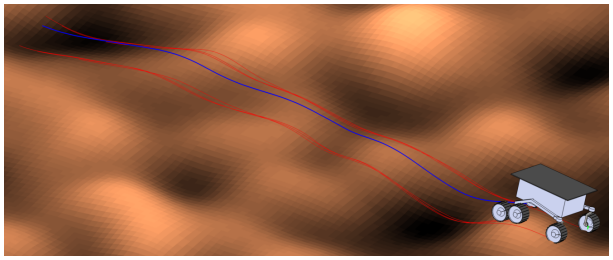
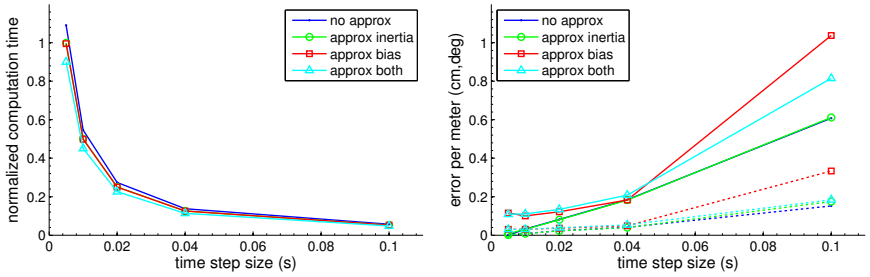


Fig. 7 Rocky 7 animation screenshot.

dom terrain while turning slightly for 10 seconds. Figure 8(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 indicates faster than real time. Each of the approximations suggested in Section 3.5 reduces computation time by approximately 10%. In Figure 8(b), error is the difference in predicted terminal pose with respect to the prediction using no approximations and the minimum (.005s) time step size, per meter of travel. Error increases linearly (not exponentially) with time step size, and only modestly with approximation of the bias force. Modest errors may be an acceptable tradeoff for significant speedup 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 constraints in a DAE formulation was demonstrated to be more stable and efficient than



(a) (b)

Fig. 8 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 yaw error.

the common method of directly adding contact forces in an ordinary differential equation formulation.

The contribution of this paper is theoretical, but physical experiments will be presented in future work on the calibration of WMR motion models. We plan to implement a WMR simulator based on this method in C++ which can be used for the fair comparison of computational cost with alternatives. We also plan to make the simulator 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 may also be fast and accurate enough for improved model-predictive planning in many 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).

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)