A Modelica Library to Add Contact Dynamics and Terramechanics to Multi-Body Mechanics

Fabian Buse[1] Antoine Pignède[1] Stefan Barthelmes[1]

[1]Institute of System Dynamics and Control (SR), German Aerospace Center (DLR), Oberpfaffenhofen Germany,
{fabian.buse,antoine.pignede,stefan.barthelmes}@dlr.de

Abstract

The *Contact Dynamics* library extends the multi-body Modelica Standard Library with contact calculation to the environment, namely soft soil and hard obstacles. A focus is on *terramechanics*, i. e. wheels driving on soft and dry soil, and a handful of models are implemented. Additionally, a Hertz contact model for hard and elastic contact, between bodies themselves or to obstacles in the environment (e. g. rocks in the soft soil), is available as well. The capabilities of the library have been key in the development of rovers for planetary exploration such as the upcoming MMX mission to the Martian moon Phobos.

Keywords: Multi-Body mechanics, Contact dynamics, Terramechanics, Modelica library with external code

1 Introduction

When developing and analyzing off-road vehicles one of the most important factors is contact dynamics, more precisely the resulting forces and torques of a metal wheel driving on unprepared soft soil. This engineering branch known as "terramechanics" has proposed numerous of models with various levels of detail describing exactly this. Starting from agricultural and military applications it has also shifted in particular to rovers that explore celestial bodies in situ, as comes clear from section 5.

Along the growing interest in planetary exploration, usage of high-fidelity modeling and simulation has also increased for development and analysis of mobile robotic systems. Modelica already provides good material for multi-body dynamics but lacks the contact detection and reaction calculation of wheels driving in soft soil or of robot parts hitting obstacles or each other. To fill this gap, the planetary exploration group at DLR's Institute of System Dynamics and Control (SR) has developed a "Contact Dynamics" library that is the subject of this text.

The text is organized as follows: section 2 reviews theory of contact dynamics and software packages for Modelica and other simulation environments. The first main section starts with the library structure and ends with some useful additions for environment and contact object generation. Details about the models themselves (idea, equations) and how they perform in simple academic examples (verification) is subject of section 4, the second main section. Finally, some example applications are shown in section 5 and concluding remarks as well as an outlook to the future are given.

All footnote links are accessed August 7, 2023.

2 State of the Art

2.1 Contact Dynamics

The mechanics of bodies in contact with each other was first scientifically inspected by Hertz (1882). Since then, the field of rigid, elastic contact has in principle not (needed to) evolved much as comes clear from Flores and Lankarani (2016) that still builds on Hertz's work. But the advent of computers and simulation has led to a large number of models for reaction force calculation based on the penetration of the bodies in contact to correctly represent the resulting speed after contact and the energy dissipated. Another fundamental contact simulation technique based on exchange of impulse also exists and is widely applied in computer games, but is not pursued further here.

The detection whether two bodies are in contact or not, has seen a few algorithms like **G**ilbert-**J**ohnson-**K**eerthi *GJK* (Gilbert et al. 1988), **P**olygonal **C**ontact **M**odel (Hippmann 2004) (not restricted to convex shapes) or **M**inkowski **P**ortal **R**efinement *MPR*, also called XenoCollide[1]. Usually, before the expensive contact detection algorithm is run, the software checks whether the **A**xis-**A**ligned **B**ounding **B**oxes *AABB* overlap to quickly exclude object pairs definitely not in contact with each other.

The Hertz contact models, more precisely the nonlinear Hunt and Crossley model as explained in Flores and Lankarani (2016), and the MPR contact detection in the open source implementation `libccd`[2], are the source for the rigid body contact dynamics in the library.

The above-mentioned contact dynamics mainly deal with two objects colliding with each other. While this also gives usable solutions for a cylinder rolling on a cuboid (a wheel driving on flat soil), better solutions for a wheel driving in soft soil are possible. This is the "terramechanics" field whose modern analysis starts with the works of M. G. Bekker and agricultural machines engineers in the 1950s and 1960s. Chapter 2 of Wong (2008) compiles the latest knowledge at the beginning of the 21st century.

[1] `http://xenocollide.snethen.com/`
[2] `https://github.com/danfis/libccd`

The highest level of detail in terramechanics is when the soil is simulated as discrete particles, each representing a small pack of grains. There is expertise also on this field at DLR SR and an in-house software `partsival` (Lichtenheldt et al. 2018). It would in theory be possible for Modelica to use `partsival` as external library, however this is not done (and won't be in the near future) for two reasons. First, discrete particles simulation requires much computer power, therefore only single wheel scenarios are possible in reasonable time. Modelica with Contact Dynamics on the other hand focus on full rover scenarios. Second, `partsival` not only simulates the particles but also the wheel dynamics themselves, thus there is no need to connect it to Modelica. A similar, divided, approach for development and analysis of the Opportunity and Spirit rovers was done with the *ARTEMIS* (Zhou et al. 2014) full rover simulation and the *COUPi* (Johnson et al. 2015) discrete particles single wheel simulations.

2.2 Contact Dynamics Simulation

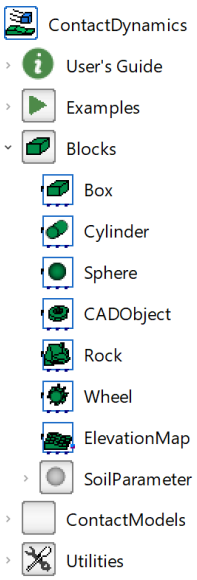
Bringing together multi-body and contact dynamics has already been done prior this work. The video game industry for example has released a dozen of *physics engines* that among other features compute contact dynamics. Emphasis there is more on visually appealing results and speed rather than on scientific accuracy and predictability of ground truth. Interestingly, the rover simulator *ROSTDyn* (Li et al. 2013) chose to code the contact dynamics themselves as a C++-library despite using the *Vortex*[3] engine that also has this capability.

Solutions for MATLAB are for example related in Tarokh (2016) and Ding et al. (2010).

Neumayr and Otter (2017) and Neumayr and Otter (2019) add collision detection and Hertz contact dynamics to *Modia3D*. The elastic, rigid body contact dynamics of the presented Contact Dynamics library share much in common with this work (MPR algorithm with AABB preprocessing, Hunt and Crossley model). However, the Contact Dynamics library goes further in that it adds a handful of contact models to Modelica. Next to the Hertz contact are specialized models for the main area of application: simulation of planetary exploration rovers, where terramechanics can be applied advantageously.

Contact dynamics for Modelica already exist as well. In fact, one can see the `ElastoGap` model of the Standard Library as a very simple contact dynamics block, as was first tried by the authors before the advent of *Contact Dynamics*. Two libraries that add contact calculation to the multi-body Standard Library, have been published so far to the knowledge of the authors. Elmqvist et al. (2015) enables Modelica for **D**iscrete **E**lement **M**ethod *DEM* using external binaries. Oestersötebier et al. (2014) uses only pure Modelica code to add punctual, linear and planar contact points to bodies, again using modern extensions of Hertz theory for the resulting forces. The free library *IdealizedContact*[4] released along this publication, is no longer maintained. Loading it into the 2022 Dymola[5] release and running the conversion script to the new Modelica Standard makes this library still usable today.

There have been a few attempts to connect Modelica to physics engines. For example, Hofmann et al. (2014) uses the collision detection capability from the *Bullet Physics*[6] as external C++-library, but Modelica for multi-body dynamics and reaction force calculations. The announced *CollisionLib* was not found by the authors. In a similar way, Bardaro et al. (2017) couples *Gazebo*[7] to Modelica. In the end, the aim of this work is more to integrate Modelica into the physics engine than to expand Modelica's capabilities with an external library.



3 Implementation

3.1 Contact Dynamics Library Overview

The base element of the Contact Dynamics library is a partial model that can be attached to any model through a multi-body frame. This base element provides a force and torque sensor, some standard parameters and small utilities such as an indicator whether a contact is present or not (this is *not* the contact detection).

Extensions from the base element come in three forms. Note that none of these blocks have mass.

- *Cuboid, cylinder, sphere, rock and CAD contact shapes:* Given parameters or path to a CAD file, the contact shape is added to the model, optionally visualized and the dynamics calculated using the BBCC or SCM contact models (or both). Primitive shapes are simulated as such while the rock asset generator subsection 3.2 is called to automatically create rocks satisfying the user's choices.

- *Wheel contact shape:* In addition to BBCC and SCM, specialized wheel-soil contact models, see subsection 4.3, can be used with this extension. The wheel asset generator subsection 3.2 is called.

- *Elevation map:* It provides two essential parts to the wheel-soil models: the geometry of the surface and the soil properties. As such it is always added as `outer` component to wheel contacts. The elevation map geometry is created using the surface asset

[3] `cm-labs.com/vortex-studio/`

[4] `github.com/oestersoetebier/IdealizedContact`

[5] `3ds.com/products-services/catia/products/dymola/`

[6] `https://pybullet.org/wordpress/`

[7] `http://gazebosim.org`

DOI 10.3384/ecp204433

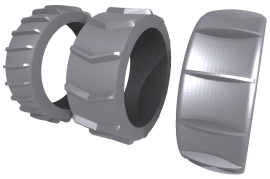


Figure 1. Examples of Wheel Generation. Left: slim with slanted grousers, center: wide with chevron grousers, right: wheel with convex curvature.

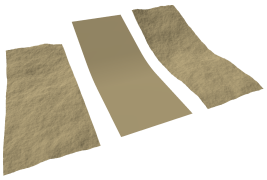


Figure 2. Examples of Surface Generation. Left: based on noise, center: based on a defined slope, right: combined.

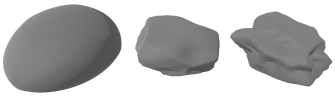


Figure 3. Examples of Rock Generation. Roughness increases from left to right.

generator subsection 3.2, the soil properties are parameters such as density and angle of repose.

The contact models are implemented independently from the contact elements. This ensures modularity and flexibility of the library. Generation of the contact shapes (cuboid, cylinder, wheel, ...) or loading from CAD is separated from the force and torque calculation too.

3.2 Assets

Part of Contact Dynamics is a procedural asset generator. This generator can generate four types of assets: wheels, surfaces, rocks and rock distributions. The procedural generation allows to create these assets based on a set of parameters and a seed. Using the same inputs will provide the same output. This feature allows the recreation of previously used assets when required.

The asset generator is a C++-library that takes the parametric definition of the various objects and provides two functions: a function to generate a unique ID for the given input and a function to generate the requested input. The Modelica interface uses these two functions to efficiently identify handles of already created objects and to create new files of not found handles. First, the ID of a requested object is generated, then the filename is defined as the combination of a type-specific prefix and the ID. Only if no file of this name already exists in the working directory, the actual wavefront `obj` file is generated and saved in the working directory with the desired name.

Wheels are generated based on an extensive parametric definition, including wheel radius, wheel width and multiple parameters defining the radial profile. This base shape can be extended by adding various features, like grousers similarly parametrized. Figure 1 shows three examples.

For surface generation, two base methods are available. The first method generates the surface based on external definitions like height maps or a profile in a single direction. The second method, procedurally generates the surface based on noise. See Buse et al. (2022) and Buse (2022) for more details on the method and validation of this noise-based terrain generation. The noise used is provided by the open source library `libnoise`[8]. These two methods can be combined: a statically defined surface based on a profile can be superimposed with a noise-based one to create more complex environments. See Figure 2 for three examples. On top of the generation of the mesh, the surface generator also allows convenient, resolution-independent access to surface information. The interface function allows access to the height and normal at given coordinates. This feature allows smooth integration with the contact models implemented in Modelica by providing the necessary information to approximate the local surface geometry into a single frame based at a given position.

Rocks are generated by deforming the surface of a sphere based on a function combining various noise types, see Buse (2022) for details. The inputs to this generation method are the average dimensions and two roughness parameters. Figure 3 shows three rocks with increasing roughness. For easier integration into a multi-body simulation, the rock generation process also computes the volume and center of gravity as well the rocks' inertia.

As rocks rarely appear alone, a generator for rock distributions combines information from the surface generator and the rock generator to create natural rock distributions. Based on a statistical distribution description, the rocks are placed on a previously generated surface. The output is either a single file including all rocks at their final position or the positions and individual files for each rock. These two options allow to either include static or dynamic rocks.

4 Contact Models in Detail

4.1 BBCC

The **B**ody to **B**ody **C**ontact model implemented in **C** *BBCC* calculates contact between two convex, rigid shapes. It is the most versatile of all models in the library and returns acceptable results in reasonable time. Special objects exist for cuboids, cylinders and spheres, a wheel is simply a cylinder with an arbitrary number of cuboids on the rim as grousers. Otherwise, a shape defined by a CAD file can be loaded, note however that the implementation of the contact detection results in effectively the convex hull being used. If wanted, collections of objects can be summarized into one "compound" object with the same properties but separate convex hulls. This is useful for scenarios with many (fixed) rocks on a surface. A surface

[8] `libnoise.sourceforge.net`

is implemented as collection of cuboids of unit height following the elevation and resolution.

The core of BBCC is an external library implemented in C that gets the position and velocity of all contact objects from Modelica and returns the resulting contact forces and torques on them. A `dll` for Windows and a shared library `so` for Linux are included as Modelica resources.

The calculation happens on a few layers. During the initialization of the simulation, a collection of objects is created in the external library containing basic information about the contact objects such as shape and size, elasticity parameters and references to other object with which collision is enabled. The Modelica BBCC shape saves this as an `ExternalObject` for later reference.

Each time instant the following is done to each object:

1. Update the position and velocity.
2. Determine whether another object is in contact:
   (a) Exclude all objects not explicitly marked as potential contact counterparts.
   (b) Exclude all objects whose AABBs do not overlap with the one of the current object.
   (c) Run the `libccd` MPR algorithm between the current and all other remaining objects.
3. Calculate the reaction forces and torques for all other objects not excluded above with the current one, see subsection 4.1 for details about the contact equations.
4. Return the sum of all forces and torques on the current object to Modelica.

As there potentially are many BBCC objects in a simulation and the order in which they're processed can't be controlled in Modelica, a special *synchronizer* is present at the top level of every model involving at least one BBCC object. It contains a connector with two variables, these are intertwined through an ordinary differential equation. This ensures that first all positions updates are sent to the external library, the contact forces are calculated only after this is done. Listing 1 shows the important code snippets for the connector, the synchronizer, the interface to the external code (`BBCC`) and the partial base class `BaseObject` from which cuboids, cylinders etc. in Modelica extend. This is similar to the synchronization idiom of Elmqvist et al. (2015). The major difference being that the external function calls are in the individual models (`BBCC`) instead of a common call in the synchronizer.

Listing 1. BBCC Synchronization

```
connector BBCC_ContactSynchronizer
  Real update;
  flow Real contact;
end BBCC_ContactSynchronizer;

model BBCC_Synchronizer
  BBCC_ContactSynchronizer sync;
initial equation
  sync.update = 0;
equation
  der(sync.update) = sync.contact;
annotation (defaultComponentName="bbccSync"
    , defaultComponentPrefixes="inner");
end BBCC_Synchronizer;

model BBCC
  BBCC_ContactSynchronizer sync;
  Real dummy;
  (...)
equation
  sync.contact = update(obj,r,v,T,w,time,
      dummy); // Update pos, vel, ...
  (force,torque) = getForce(obj,time,sync.
      update); // Calculate f, tau
  (...)
end BBCC;

partial model BaseObject
  outer BBCC_Synchronizer bbccSync;
  BBCC bbcc(dummy=bbccSync.dummy,...);
  (...)
equation
  connect(bbccSync.sync, bbcc.sync);
  (...)
end BaseObject;
```

4.1.1 Model

The normal force on objects in contact is (Flores and Lankarani 2016)

$$\|F_N\| = k\delta^{1.5}(1+d\dot{\delta}) \tag{1}$$

$$d = \frac{8}{5}\frac{1-\zeta}{\zeta\dot{\delta}^{(-)}} \tag{2}$$

$$k = \frac{4}{3}\frac{\sqrt{R}}{0.5\left(\frac{1-\nu_1^2}{E_1}+\frac{1-\nu_2^2}{E_2}\right)} \tag{3}$$

with penetration depth δ, relative penetration velocity $\dot{\delta}$ and relative impact velocity prior to contact $\dot{\delta}^{(-)}$. The resulting stiffness k and damping d are functions of the object parameters coefficient of restitution harmonic mean $\zeta = 2\frac{\zeta_1\zeta_2}{\zeta_1+\zeta_2}$, modulus of elasticity E and Poisson number ν. R is an estimate of the effective Hertz contact radius, currently estimated as half the harmonic mean of the longest edges of the objects in contact ($\frac{1}{R} = \frac{1}{R_1}+\frac{1}{R_2}$). Note also that the intuitive meaning of the coefficient of restitution (ratio between pre- and post-impact velocity) is lost in this algorithm. In general $\zeta \neq \frac{\dot{\delta}^{(+)}}{\dot{\delta}^{(-)}}$, although this relationship is at the basis of the normal force derivation.

The default tangential contact model is

$$\|F_T\| = \mu\|F_N\|\tanh\left(\frac{\|v_T\|}{v_d}\right) \tag{4}$$

representing the Coulomb friction regularized by the velocity dead band (lower bound for velocities) parameter v_d, with $\mu = 2\frac{\mu_1\mu_2}{\mu_1+\mu_2}$ being the harmonic mean of the friction parameters of the objects in contact.

DOI
10.3384/ecp204433

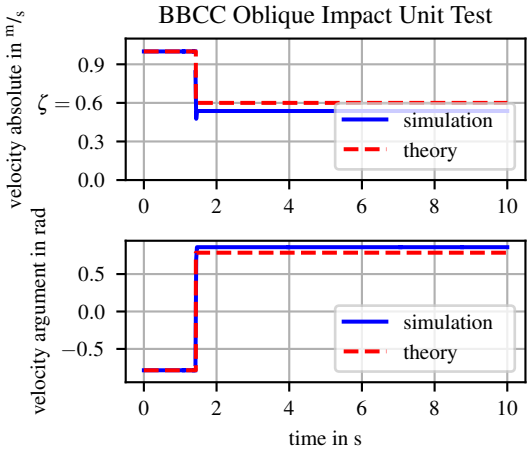


Figure 4. BBCC Impact Verification: Cylinder against cylinder in zero gravity

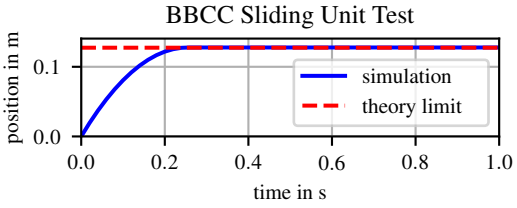


Figure 5. BBCC Sliding Verification: Non-rolling cylinder with initial velocity on flat plane

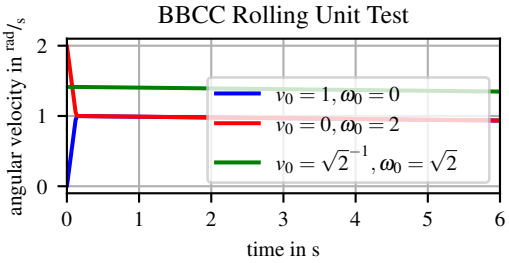


Figure 6. BBCC Rolling Verification: Free to roll cylinder with initial energy differently distributed on flat plane

The user can provide two more parameters μ_s and ξ to activate a static tangential force model after Bengisu and Akay (Marques et al. 2016) which is stiction capable

$$\|F_T\| = \begin{cases} \|F_N\|\mu_s - \frac{\|F_N\|\mu_s}{v_d^2}(\|v_T\| - v_d)^2 & \forall \|v_T\| < v_d \\ \|F_N\|\mu + \|F_N\|(\mu_s - \mu)e^{-\xi(\|v_T\| - v_d)} & \forall \|v_T\| \geq v_d \end{cases} \tag{5}$$

The torque in normal direction reads

$$\|\tau_N\| = \mu\|F_N\|\pi\sqrt{R\delta}^2 \tanh\left(\frac{\sqrt{R\delta}\|\omega\|}{vd}\right) \tag{6}$$

with the effective contact area $\sqrt{R\delta}$. There (currently) is no tangential torque in BBCC.

4.1.2 Verification

This text is the first publication that goes into detail about this contact model. Thus, this extensive verification subsection to prove the correctness of BBCC in academic examples and give hints about the performance in more practical applications, as those of section 5. The tests were done on a Windows 10 computer with Dymola 2022 and the `Esdirk45a` solver (tolerance 1×10^{-5}). The parameters used for all contact objects are given in Table 1.

Table 1. BBCC Verification Cylinder Parameters

Description	*Symbol*	*Value*
Radius	r	0.50 m
Height	h	1.00 m
Mass	m	1.00 kg
Inertia	I	0.25 kg m^2
Young	E	4.50×10^5 Pa
Poisson	ν	0.40
Restitution	ζ	0.60
Friction	μ	0.40

Definitions and results of the tests to verify BBCC:

- *Impact:* The wheel is initialized above the surface with an initial velocity oblique to the flat surface. Gravity is turned off and the expected outcome is that the incidence angles match.
  Figure 4 shows that the BBCC cylinder correctly applies the rules, that the incidence angle absolute value after impact is the same than before, that the velocity absolute value is reduced by the ζ, within an acceptable tolerance.
  Tests in Earth gravity with $\zeta = 1.0$ or 0.0 (full energy conservation or dissipation) verify successfully.

- *Sliding:* The wheel is initialized with an initial velocity tangential to the flat surface, the wheel can't rotate but must slide. The expected outcome is that the motion is slowed down because of friction.
  Figure 5 shows that the BBCC friction breaking of a cylinder with initial velocity on a cuboid is correct.

- *Rolling:* The wheel is initialized with an initial velocity tangential to the flat surface, the wheel can rotate. The expected outcome is that the motion is not slowed down (much) because of friction.
  The initial translational and rotational velocities of Figure 6 were chosen to have the same initial energy in all three cases. For the translational (blue) respectively rotational (red) initial energy only, conversion into matching rotational and translational velocities dissipates some energy compared to the case of equal initial energies (green). Following this, little losses due to rolling friction are seen with the same constant deceleration in the three cases.

- *Sliding on inclined plane:* The wheel is initialized with zero initial velocity on a flat, inclined surface, the wheel can't rotate but must slide. The expected

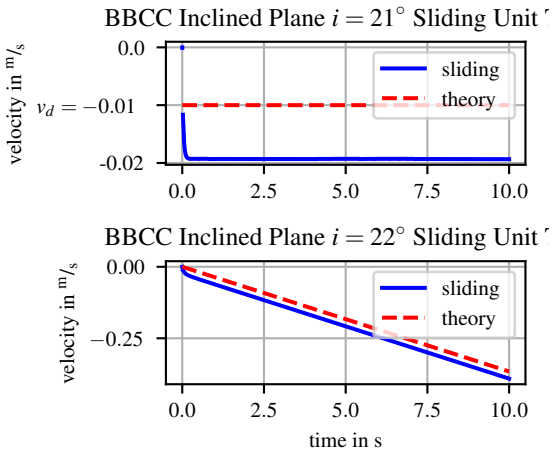


Figure 7. BBCC Sliding Verification on Inclined Plane: Non-rolling cylinder on inclined planes, $i \leq \arctan \mu$ or $i > \arctan \mu$

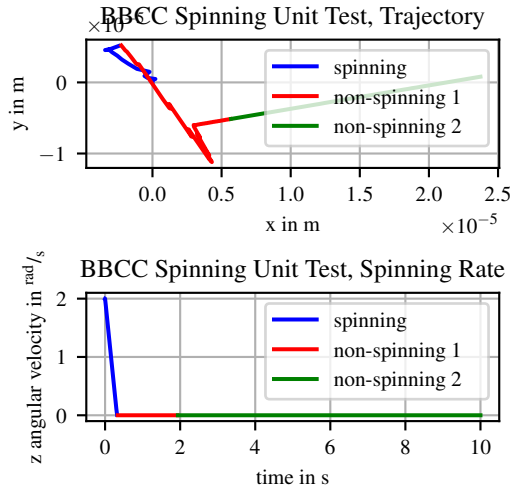


Figure 8. BBCC Spinning Verification: Cylinder with initial normal angular velocity on flat plane, the line in the plots is divided into three segments for easier understanding

outcome is that the motion is either zero (low inclination) or accelerated (high inclination).
If the inclination is *smaller* than the corresponding friction angle $\arctan \mu$ the cylinder correctly slides down the plane with constant velocity as visible in the upper plot of Figure 7. This velocity is higher than the expected v_d of Equation 4 because of the short time needed for the contact to be settled after initialization. If the inclination is *higher* than the corresponding friction angle $\arctan \mu$ the cylinder correctly slides down the plane with constant acceleration as visible in the lower plot of Figure 7. The velocity is higher than expected because of the short time needed for the contact to be settled after initialization, the acceleration though is correct.
If the cylinder can *roll down the inclined plane*, the expected constant acceleration regardless whether $i \leq \arctan \mu$ or $i > \arctan \mu$ is correctly simulated.

- *Spinning:* The wheel is initialized with an angular velocity normal to the flat surface. The expected outcome is constant deceleration because of friction. Figure 8 shows the correct almost complete energy dissipation. As the (x, y)-trajectory in the upper plot shows, a very little constant rest velocity remains.

The points above show the successful verification of the contact model against academic examples within an acceptable tolerance. This doesn't rule out unwanted behavior though, e. g. when stacking cylinders on top of each other on their mantle sides. The instability of the equilibrium points quickly brings the tower to fall. This effect can also be seen in stable equilibrium, e. g. stacking of cuboids. There, the contact points jump between corners of the same face and the cuboids never come completely to rest. Still, the movements remain small enough for practical time spans such that a cuboid tower doesn't fall.

4.2 SCM

The **S**oil **C**ontact **M**odel *SCM*, first presented in Krenn et al. (2008) and significantly advanced in recent years (Buse 2018; Buse 2022), aims to provide detailed terramechanical modeling in a form suitable for multi-body simulations. Explicitly modeling soil deformation SCM, or the newer version FSCM (*Flow based* Soil Contact Model), allows effects like ruts left behind by wheels or other permanent soil deformation to affect system behavior. The SCM model is a self-contained C++-library. The interface code contained in Contact Dynamics provides integration with Modelica and the DLR Visualization 2 Library.

The interface to SCM is divided into two main parts, SCM contact objects and SCM surfaces. SCM contact objects define geometries that can interface with the surface. Each object is defined by a mesh representing the geometry, position, velocity, orientation and angular velocity determining the current pose. For each of these objects, SCM provides the resulting reaction forces. The SCM surface defines the regolith surface, it is a stationary object which describes the surface geometry as well as parameters. Internally a horizontal equidistant grid is used, and the height of each node in the grid is used to represent the geometry. The Contact Dynamics blocks of type box, cylinder, sphere, CAD, rock and wheel represent the SCM contact objects. As SCM always relies on a mesh to represent the contact geometry, base meshes for a box, cylinder and sphere were manually created and placed in the library resources. These are then scaled to match the desired dimensions. For more complex shapes like rock and wheel, custom meshes can be generated, see subsection 3.2. The SCM surface is part of the elevation map block. An example of a single-wheel driving through soft regolith is shown in Figure 9.

SCM divides contact modeling into two functions: a contact dynamics function and a soil update function. The contact dynamics function is called once for each object and timestep, it computes the reaction forces based on the current object pose and the last known soil state. In

DOI
10.3384/ecp204433

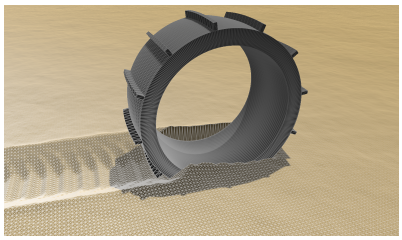


Figure 9. SCM Visualization: A wheel driving through soft soil. The grid representation of the surface is shown as wireframe.

the soil update function, the surface geometry and internal soil states are updated based on the last known object positions. The soil deformation results in a change of node height in the surface grid. These two functions are coupled to two sampled clocks in Modelica to control the rate at which these functions are executed independently.

To visualize the surface, a flexible surface from the DLR Visualization 2 Library is directly integrated with the SCM library, this allows a higher resolution compared to an integration through Modelica. This allows the visual representation of detailed rutting as shown in Figure 9.

4.2.1 Verification

Extensive validation of SCM is documented in (Buse 2022). In this campaign, the model's surface deformation and force prediction has been compared against measurements taken with the **T**erramechanics **Ro**botics **L**ocomotion **L**ab *TROLL*. This testbed allows automatic testing of various terramechanical experiments, see (Buse 2019). Figure 10 shows data from one scenario performed in the validation. SCM's predicted traction, normal and drive torque are compared with the measurements when the wheel is moved along the trajectory captured by the testbed. In the shown scenario, a wheel is placed on a flat surface and then vertically loaded to 100 N. After a short period, a movement combining a translational movement of the robot and a wheel rotation is started. A slip ratio of 60 % is enforced during this movement, thus the translational velocity is only 40 % of what the wheel's rotational velocity and radius would suggest.

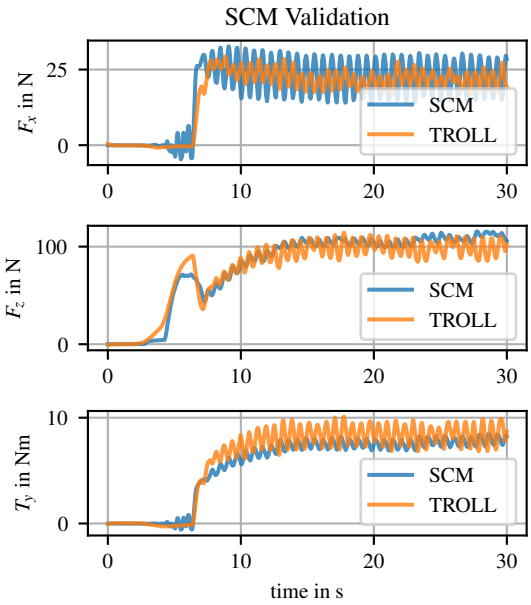


Figure 10. SCM single wheel comparison of normal force, traction force and drive torque. Terramechanics Robotics Locomotion Lab measurements in orange, simulated forces and torque when replaying the motion captured with the testbed in blue.

4.3 Terramechanical Wheel-Soil Models Directly Implemented in Modelica

These models are not general like BBCC, they are only valid for a wheel driving in soft soil.

The wheel contact shape, see subsection 3.1, contains a wider selection of models to calculate the contact dynamics than cuboids, cylinders and spheres. In contrast to the BBCC and SCM models that extend `oneFrame` multi-body interfaces, these other conditional models extend from `twoFrame`. The frame on the left-hand side is used again to get the position and orientation of the wheel and receives the reaction forces and torques. The right-hand side frame gets no forces but is connected to the elevation map and is used to detect whether the wheel is in contact, remember that contact detection is external in SCM and BBCC. Creation of elevation maps is explained in subsection 3.2. The equations governing the models are detailed in the following.

4.3.1 TerRA

TerRA is short for **Ter**ramechanics for **R**eal-time **A**pplications and is a purely empirical, fast computing terramechanics model developed by Barthelmes (2018). The scope of this model is to provide a model that captures the main effects of wheel-soil interaction while still being considerably faster than real-time to allow using it in onboard control software. The model captures dynamic slip-sinkage and its effects on the traction and resistance forces while not using any spatial discretization. TerRA consists of purely empirical relations and its parameters are not derived from any physical soil properties. They thus need to be tuned with a higher fidelity model or experimental data with the help of an optimization algorithm.

Model One very important effect in wheel-soil interaction is dynamic sinkage: A wheel sinks deeper into the soil for higher slippage and climbs out of its ditch once traction is sufficient to reduce slip. Typically, modeling these effects is either done with a spatial discretization of the ground to consider the sinkage as a position-dependent state or by altering the normal stress or normal force with the wheel slip. In TerRA a more explicit approach is developed, where the overall slip is divided into a sinkage that is effective for increasing traction and the slip-sinkage. The difference can be more easily understood when imagining a wheel with large grousers: If the wheel is pushed into the soil by normal force while not rotating, the soil below the wheel is mostly compressed, which

increases the traction potential once the wheel starts rotating. If the wheel rotates but does not move forward, the grousers shovel the soil away, leading to sinkage of the wheel as well, however, the soil below the wheel is removed rather than compressed, which increases the traction potential much less.

In TerRA, the total sinkage is therefore composed of the effective and the slip-sinkage

$$z = z_{\text{eff}} + z_{\text{s}}. \tag{7}$$

The slip sinkage is calculated with a dynamic model from the inward and outward dynamic sinkage as

$$\dot{z}_{\text{s}} = \dot{z}_{\text{in}}(\omega r - v_x) + \dot{z}_{\text{out}}(z_{\text{eff}}, z_{\text{s}}, v_x, \alpha) \tag{8}$$

where $\omega r - v_x$ is the slip velocity, v_x the forward velocity and α the ground inclination. The exact relations contain several wheel/robot, soil and model parameters that result in a roughly exponential function that can be shaped for different soil and wheel types.

For the traction force, a maximum shear length is calculated by integrating the slip velocity over time. To consider changing conditions and new, unsheared, soil coming into contact, TerRA uses a special additional state that moves along the contact patch depending on the slip vs. forward velocities. While the standard Janosi-Hanamoto relations assume the shear length as a wheel state, the additional state in TerRA accounts for the fact that mainly the soil shear length results in traction potential.

Finally, a resistance force is calculated based on passive Earth pressure and therefore dependent on total sinkage.

Verification TerRA has several parameters that cannot be derived directly from physical properties of the wheel and soil. Therefore, the qualitative behavior as well as the tunability to the high-fidelity SCM model were investigated in Barthelmes (2018). The dynamic sinkage behavior of SCM can be replicated with TerRA to an error of less than 3 %, however, considerable differences remain especially in the dynamic drawbar pull force development.

Recently, TerRA is being used as one of three models of different fidelity for teaching a machine learning terramechanics model (Fediukov et al. 2022). Within this work, a better fitting between SCM and TerRA was achieved.

4.3.2 Other Terramechanics Models

Similar to TerRA, three other terramechanical models are implemented directly in Modelica, these can only be used for wheel contact shapes to a non-deformable surface. This subsection is only a short summary, because of the minor importance of these models in the Contact Dynamics library, interested readers are referred to Lichtenheldt et al. (2016) for a more elaborate discussion.

Two models attempt to implement Bekker's terramechanics equations following Chapter 2 in Wong (2008). Depending on the actual approach chosen ("pure" Bekker or Bekker-Janosi-Hanamoto) some differences in details and behavior are introduced, the parameters are also derived differently from wheel and surface (geometry and soil). But in the end, both are implementations directly in Modelica to compute the reaction forces on a wheel in a fast and easy way. Hence, precision and fidelity are low. Still, some basic effects in academic examples (sliding on an inclined plane) can be reproduced as expected.

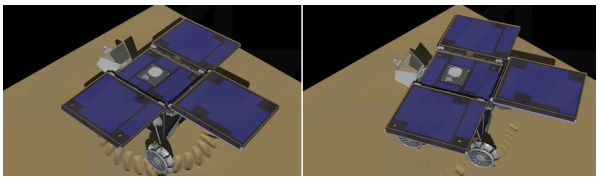


Figure 11. MMX Rover Point Turn Simulation on SCM Soil with Cohesion 20 Pa and 200 Pa

A third low fidelity, "rheological", terramechanics model for wheel to surface contact is implemented directly in Modelica as well. This one sees the ground as a spring-damper system and again derives parameters from wheel and soil properties. One interesting part of this model is the stiction capability, which is implemented using control logic elements: a PID-controller regulates the wheel frame to rest as long as the stiction force is not overcome, or if the wheel reaches a lower speed limit. The same academic tests as in the two Bekker models can be reproduced.

5 Applications

The Contact Dynamics library was originally completely integrated into the DLR **R**over **S**imulation **T**oolkit *RST* (Hellerer et al. 2017) but soon was extracted as standalone library. Planetary exploration rovers however, remain the main area of applications. RST essentially extends the contact blocks with rigid bodies from the Standard Library and adds further domains such as power and control logic. Two examples where DLR SR has applied the Contact Dynamics library are detailed in the following, with an emphasis on the contact dynamics and how the library has been key for these projects.

5.1 MMX

The **J**apan **A**erospace e**X**ploration **A**gency *JAXA* is in the preparation of a mission to the moons of Mars with sample return. This mission, known as **M**artian **M**oons e**X**ploration *MMX*, is carrying a rover jointly developed by the German DLR and French CNES to explore Phobos in situ (Ulamec et al. 2021). As detailed in Buse et al. (2022), simulations using the Contact Dynamics library were an integral part of the development process and will also be important for the operations and analysis phases.

Specifically, many mission phases were simulated with SCM for the contact of the wheels or other rover parts to the surface and BBCC for contact to rocks or between rover parts. Phobos is not completely unknown like comets and asteroids on first encounter, thus there are plausible number ranges about the topography and rock

DOI
10.3384/ecp204433

distribution. In fact, the asset generators for rocks and surface (subsection 3.2) were designed with these planetary science data in mind. Without the Contact Dynamics library, it would not have been possible to verify the sequence of movements to deploy the rover from its stowed configuration after landing on Phobos.

Contact Dynamics is also important to estimate driving performance. For example, Figure 11 shows the simulation of a point turn (90° if there would be no slip) on SCM soil with different cohesion values. Once the rover will have driven on Phobos and returned telemetry, validation of the contact model for milli-gravity will be undertaken.

The other contact models are not used for MMX.

5.2 Scout

DLR's institute of SR is currently developing a small, modular and highly agile rover for extreme terrain and cave exploration called *Scout* (Lichtenheldt et al. 2021). The team follows paradigms such as rapid control prototyping and model-based development (Pignède et al. 2022), simulation plays an important role and contact dynamics are central to the results, justifying the term "simulation driven development".

For example, the stiffness in the backbone was adjusted after an extensive simulation campaign where the rover was sent through an obstacle parcours with stairs, slopes etc. (Pignède and Lichtenheldt 2022). The BBCC model was used. A challenge in this activity is the huge number of objects. The nominal Scout rover consists of three modules (cuboid contact objects) with two wheels each. Each wheel has three spokes with an arc-like form simplified to two cuboids, and a foot at the end approximated as cylinder. This sums up to 57 contact objects for the rover to which 38 cuboids are added of the obstacle parcours, see Figure 12. Here, the BBCC capability to group objects into collections of pairs that are not tested for collision, were essential to keep the simulation time reasonable. Also, the test for overlapping axis-aligned bounding boxes filters out many pairs before calling the computationally expensive proper collision detection.

The simulation also serves to test new software before setting the prototype to risk. As low precision and fast simulation is often required, one of the simple terramechanics models of subsubsection 4.3.2 is used, with wheel contact objects of appropriate parameters as feet.

6 Conclusion and Further Work

The DLR *Contact Dynamics* library provides various types of contact dynamics to Modelica multi-body mechanics. It focuses especially on terramechanics for development and analysis of planetary exploration rovers but also includes two general models for contact between rigid, elastic bodies. Generators for environment and wheels are also included in the package. The structure and implementation of the library permits diverse applications at various levels of detail to assist engineers in all phases of projects from inception to post processing of field data. It's an integral part of the SR's toolchain for modeling, simulation and optimization of planetary exploration rovers and beyond, projects such as the MMX and Scout rovers much rely on Contact Dynamics.

This text has presented the library in general with more detailed sections about previously unpublished material. Simple models of the library are implemented directly in Modelica, more advanced ones are included as external code. These have been verified against ground truth (SCM), models of higher fidelity (TerRA) or academic examples (BBCC) to ensure validity of results generated using the library. The asset generator is a feature, unique in the Modelica world, to create random environments that meet statistical properties automatically.

Although the library is in good use already today, some tasks for further work remain. Currently only SCM has been validated against ground truth, BBCC should also go through this process. Verification of the models will be extended to milli-gravity environment using data collected by the MMX rover on Phobos. There also is potential to increase simulation speed with BBCC using multithreading as is already done with SCM.

Acknowledgements

The authors thank T. Bellman and his team, especially M. Hellerer, S. Kümper, R. Reiser and M. Neves.

References

Bardaro, Gianluca et al. (2017). "Using Modelica for advanced Multi-Body modelling in 3D graphical robotic simulators". In: *12th International Modelica Conference, Prague, Czech Republic*. Linköping University Electronic Press, pp. 887–894. DOI: 10.3384/ecp17132887.

Barthelmes, Stefan (2018). "TerRA: Terramechanics for Real-time Application". In: *5th Joint International Conference on Multibody System Dynamics, Lisbon, Portugal*. Springer.

Buse, Fabian (2018). "Using superposition of local soil flow fields to improve soil deformation in the DLR Soil Contact Model - SCM". In: *5th Joint International Conference on Multibody System Dynamics, Lisbon, Portugal*. Springer.

Buse, Fabian (2019). "Fully Automated Single Wheel Testing with the DLR Terramechanics Robotics Locomotion Lab (TROLL)". In: *15th Symposium on Advanced Space Technologies in Robotics and Automation, Noordwijk, Netherlands*. ESA.

Buse, Fabian (2022). "Development and Validation of a Deformable Soft Soil Contact Model for Dynamic Rover Simulations". Dissertation. Tohoku University.

Buse, Fabian et al. (2022). "MMX Rover Simulation - Robotic Simulations for Phobos Operations". In: *2022 IEEE Aerospace Conference, Big Sky, MT, USA*. IEEE. DOI: 10.1109/AERO53065.2022.9843391.

Ding, Liang et al. (2010). "Terramechanics-based high-fidelity dynamics simulation for wheeled mobile robot on deformable rough terrain". In: *2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA*. IEEE, pp. 4922–4927. DOI: 10.1109/ROBOT.2010.5509217.

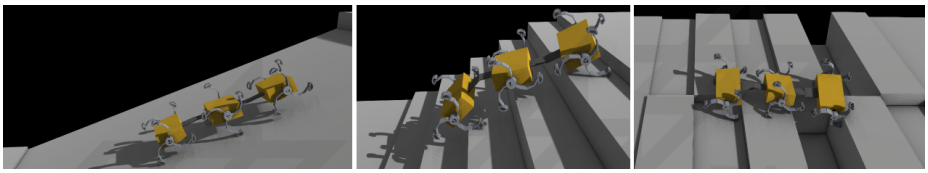


Figure 12. Scout Rover Obstacle Parcours Simulation with BBCC Contact Dynamics

Elmqvist, Hilding et al. (2015). "Generic Modelica Framework for MultiBody Contacts and Discrete Element Method". In: *11th International Modelica Conference, Versailles, France*. Linköping University Electronic Press. DOI: 10 . 3384 / ecp15118427.

Fediukov, Vladyslav et al. (2022). "Multi-Fidelity Machine Learning Modeling for Wheeled Locomotion on Soft Soil". In: *11th Asia-Pacific Regional Conference of the ISTVS, Harbin, China*. ISTVS. DOI: 10.56884/WGPV6693.

Flores, Paulo and Hamid M. Lankarani (2016). *Contact Force Models for Multibody Dynamics*. Springer. ISBN: 978-3-319-30897-5.

Gilbert, Elmer G. et al. (1988). "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space". In: *IEEE Journal of Robotics and Automation* 4.2, pp. 193–203. DOI: 10.1109/56.2083.

Hellerer, Matthias et al. (2017). "The DLR Rover Simulation Toolkit". In: *14th Symposium on Advanced Space Technologies in Robotics and Automation, Leiden, Netherlands*. ESA.

Hertz, Heinrich (1882). "Ueber die Berührung fester elastischer Körper". In: *Journal für die reine und angewandte Mathematik* 1882.92, pp. 156–171. DOI: 10.1515/crll.1882.92.156.

Hippmann, Gerhard (2004). "Modellierung von Kontakten komplex geformter Korper in der Mehrkorperdynamik". Dissertation. Technische Universität Wien.

Hofmann, Andreas et al. (2014). "Simulating Collisions within the Modelica MultiBody Library". In: *10th International Modelica Conference, Lund, Sweden*. Linköping University Electronic Press, pp. 949–957. DOI: 10.3384/ECP14096949.

Johnson, Jerome B. et al. (2015). "Discrete element method simulations of Mars Exploration Rover wheel performance". In: *Journal of Terramechanics* 62, pp. 31–40. DOI: 10.1016/j.jterra.2015.02.004.

Krenn, Rainer et al. (2008). "Contact Dynamics Simulation of Rover Locomotion". In: *9th International Symposium on Artificial Intelligence, Robotics and Automation in Space Los Angeles, CA, USA*. ESA.

Li, Weihua et al. (2013). "ROSTDyn: Rover simulation based on terramechanics and dynamics". In: *Journal of Terramechanics* 50.3, pp. 199–210. DOI: 10.1016/j.jterra.2013.04.003.

Lichtenheldt, Roy et al. (2016). "Wheel-Ground Modeling in Planetary Exploration: From Unified Simulation Frameworks Towards Heterogeneous, Multi-tier Wheel Ground Contact Simulations: Chapter 8". In: *Multibody Dynamics*. Ed. by Josep M. Font-Llagunes. Springer, pp. 165–192. ISBN: 978-3-319-30614-8.

Lichtenheldt, Roy et al. (2018). "partsival – Collision-based Particle and many-body Simulations on GPUs for Planetary Exploration Systems". In: *5th Joint International Conference on Multibody System Dynamics, Lisbon, Portugal*. Springer.

Lichtenheldt, Roy et al. (2021). "A Mission Concept For Lava Tube Exploration On Mars And Moon – The DLR Scout Rover". In: *52nd Lunar and Planetary Science Conference*. LPI.

Marques, Filipe et al. (2016). "On the Frictional Contacts in Multibody System Dynamics: Chapter 4". In: *Multibody Dynamics*. Ed. by Josep M. Font-Llagunes. Springer, pp. 67–91. ISBN: 978-3-319-30614-8.

Neumayr, Andrea and Martin Otter (2017). "Collision Handling with Variable-Step Integrators". In: *8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, Weßling, Germany*. ACM, pp. 9–18. ISBN: 9781450363730.

Neumayr, Andrea and Martin Otter (2019). "Collision handling with elastic response calculation and zero-crossing functions". In: *9th International Workshop on Equation-based Object-oriented Modeling Languages and Tools, Berlin, Germany*. ACM, pp. 57–65. DOI: 10.1145/3365984.3365986.

Oestersötebier, Felix et al. (2014). "A Modelica Contact Library for Idealized Simulation of Independently Defined Contact Surfaces". In: *10th International Modelica Conference, Lund, Sweden*. Linköping University Electronic Press, pp. 929–937. DOI: 10.3384/ECP14096929.

Pignède, Antoine et al. (2022). "Toolchain for a Mobile Robot Applied on the DLR Scout Rover". In: *2022 IEEE Aerospace Conference, Big Sky, MT, USA*. IEEE. DOI: 10 . 1109 / AERO53065.2022.9843816.

Pignède, Antoine and Roy Lichtenheldt (2022). "Modeling, simulation and optimization of the DLR Scout rover to enable extraterrestrial cave exploration". In: *6th Joint International Conference on Multibody System Dynamics, 10th Asian Conference on Multibody Dynamics, New Delhi, India*. Springer.

Tarokh, Mahmoud (2016). "Kinematics-Based Simulation and Animation of Articulated Rovers Traversing Rough Terrain". In: *2016 International Conference on Modeling, Simulation and Visualization Methods, Las Vegas, NV, USA*. WORLDCOMP, pp. 3–9.

Ulamec, Stephan et al. (2021). "The MMX Rover Mission to Phobos: Science Objectives". In: *72nd International Astronautical Congress, Dubai, UAE*. IAF.

Wong, J. Y. (2008). *Theory of ground vehicles*. 4th. John Wiley & Sons, Inc. ISBN: 9780470170380.

Zhou, Feng et al. (2014). "Simulations of Mars Rover Traverses". In: *Journal of Field Robotics* 31.1, pp. 141–160. DOI: 10.1002/rob.21483.