Exomars Rover Mechanical Modeling with Siconos

Jan Michalczyk, Maurice Brémond, Vincent Acary, Roger Pissard-Gibollet

Exomars Rover Mechanical Modeling with Siconos

Jan Michalczyk, Maurice Brémond, Vincent Acary, Roger Pissard-Gibollet

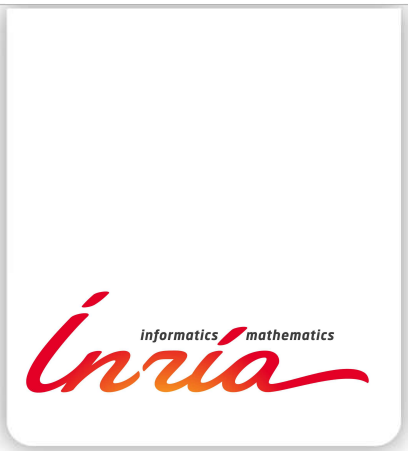
TECHNICAL REPORT

N° 448

July 2014

Project-Team Bipop

ISSN 0249-0803 ISRN INRIA/RT--448--FR+ENG



Exomars Rover Mechanical Modeling with Siconos

Jan Michalczyk, Maurice Brémond, Vincent Acary, Roger Pissard-Gibollet

Project-Team Bipop

Technical Report n° 448 — July 2014 — 38 pages

Abstract: This document contains specification and documentation of the tests performed on the exomars planetary rover model using Siconos software. Siconos is an open source scientific software targeted at modeling and simulating nonsmooth dynamical systems. The advantage of using Siconos for mechanical simulations is that it offers efficient friction-contact nonsmooth models. Performing mechanical simulations of the exomars rover appears to be of great importance with respect to verifying its design and estimating energy consumption. Seven different scenarios have been defined in order to simulate the mechanical behaviour of the exomars rover. Dynamics and kinematics models of the exomars rover have been created using HuMAnS software and implemented inside Siconos using its C code generator as plugin functions. The intention of this report is to create a common reference and unification of mechanical tests of the exomars rover across different simulation platforms.

Key-words: robotics, dynamical systems, nonsmooth dynamics, multibody systems, simulation

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Modélisation Mécanique du Rover Exomars avec Siconos

Résumé : Ce document contient les spécifications et la documentation des tests mécaniques du rover martien exomars executés à l'aide de Siconos. Siconos est une platforme scientifique open source visée pour la simulation des systémes dynamiques nonrégulières. L'avantage de l'utilisation de Siconos est son modèle efficace des problèmes de type friction-contact. La Simulation mécanique du rover Exomars est primordiale pour la vérification de sa conception et l'éstimation de la consommation de l'énergie. Sept scénarios différents ont été définis afin de simuler le comportement mécanique du rover. Les modèles dynamiques et cinématiques du rover ont été crées à l'aide d'un outil HuMAnS et implémentés en Siconos grâce au générateur du code C de HuMAnS et le mécanisme des plugins de Siconos. L'objet de ce rapport est la création d'une référence commune et l'unification des tests mécaniques du rover exomars à travers les différentes platformes de simulation.

Mots-clés : robotique, systèmes dynamiques, dynamique nonrégulière, systèmes multicorps, simulation

1 Introduction

This document contains the specification and documentation of tests performed on the ESA (European Space Agency) Exomars rover model using Siconos software. It can also be considered an expansion and complement of work outlined in [2]. Siconos is an open source software code capable of dealing with idealized contact interaction based on unilateral (rigid) contact, 3D Coulomb friction and impacts. Using such software in planetary rover's simulations may help in designing control laws for the locomotion and simulating the global behavior of the system. Model of the system has been created using HuMAnS software and implemented using its C code generator. The intention thereof is a common reference and unification of mechanical tests of the Exomars rover across different platforms. The document is structured in such way that firstly some basics on the Nonsmooth Mechanics and Siconos are given, then motivations for using the Nonsmooth approach are laid down and finally the first chapter ends with two subsections on how the simulations have been set up in Siconos and how the model of the system has been obtained. Second chapter contains detailed test cases along with comments on each of them.

1.1 Nonsmooth Mechanics at a glance

In this section, we recall the main ingredients of the nonsmooth mechanics approach for the modeling and the simulation of mechanical systems with contact and friction. For more details, we refer to [3]. Let us start with the following Lagrangian formulation of the equations of motion

$$\begin{cases} \dot{q} = v, & \text{(1a)} \\ M(q)\dot{v} + N(q,v) + F(t,q,v,u) = G^\top(q)\lambda, & \text{(1b)} \\ \dot{u} = d(t,q,v,u,\lambda), & \text{(1c)} \\ g_k(q) = 0, \quad k \in \mathcal{E} & \text{(1d)} \\ g_k(q) \geq 0, \quad \lambda_k \geq 0, \quad \lambda_k g_k(q) = 0 \quad k \in \mathcal{I}, & \text{(1e)} \end{cases}$$

where $q(t) \in I\!R^n$ is a set of coordinates that gives in a unique way the configuration of the system. The vector $v(t) \in I\!R^n$ describes the velocity. The matrix $M \in I\!R^{n\times n}$ is a symmetric, positive and (semi-)definite matrix representing the inertia. The vector of gyroscopic effects is denoted as $N \in I\!R^n$. The vector of forces $F \in I\!R^n$ applied to the system includes the internal and external forces and the effect of the vector $u \in I\!R^{n_u}$ which might be considered as a control input vector. The dynamics of this control can be described by another first order dynamical system (1c). The sets $\mathcal{E} \subset I\!N$ and $\mathcal{I} \subset I\!N$ are the index sets of bilateral constraints ($\mathcal{E}$ stand for equalities) and of *unilateral constraints* ($\mathcal{I}$ stand for inequalities), respectively. The matrix $G(q) = \nabla_q^\top g(q)$ is the Jacobian of the constraints. The condition (1e) is called a *complementarity condition*, or equivalently the *Signorini condition*. It can be equivalently written as

$$0 \leq g_k(q) \perp \lambda_k \geq 0 \quad \text{for all } k \in \mathcal{I}. \tag{2}$$

This condition models the basic unilateral contact condition. The complementarity comes from the fact that the reaction force at contact represented by λ can only act when the contact is closed, that is when $g_k(q) = 0$.

In order to complete the model at contact, let us introduce the local contact variables. Usual kinematic relations between the local velocity $U_k \in I\!R^3$ at one contact k and the generalized variables, and by duality, the generalized forces r due to contact forces $R_k \in I\!R^3$ give

$$U_k(t) = G_k(q)\dot{q} = G_k(q)v, \qquad r = G_k^\top(q)R_k. \tag{3}$$

Let us drop the subscript k to lighten the notation. By introducing a local frame at contact $(\mathrm{N}, \mathrm{T}, \mathrm{S})$ where N is a unit normal vector, we can decompose the local variables as $U = U_{\mathrm{N}}\,\mathrm{N} + U_{\mathrm{T}}, U_{\mathrm{T}} \in I\!R^2, \quad R = R_{\mathrm{N}}\,\mathrm{N} + R_{\mathrm{T}}, R_{\mathrm{T}} \in I\!R^2$. The Lagrangian dynamics in (1) is usually not so smooth. If the normal relative velocity at contact $U_{\mathrm{N}}(t^\star)$ is positive when the bodies hit at time $t^\star$, in other words, when an impact occurs, a velocity jump must occur to satisfy the constraints after $t^\star$ and an impact law has to be added. Let us simply choose the Newton impact law

$$U_{\mathrm{N}}^+(t^\star) = -eU_{\mathrm{N}}^-(t^\star), \text{ for all } t \text{ such that } g(q(t^\star)) = 0. \tag{4}$$

By formulating the unilateral contact in terms of local variables at contact we get the impact law at the velocity level

$$\text{if } g_{\text{N}} \triangleq g(q) \leq 0, \quad \text{then } 0 \leq U_{\text{N}}^{+} + eU_{\text{N}}^{-} \perp R_{\text{N}} \geq 0. \tag{5}$$

Coulomb's friction can be expressed in a disjunctive form as

$$\text{if } g_{\text{N}} \leq 0, \begin{cases} \text{if } U_{\text{T}} = 0 & \text{then } R \in \mathbf{C} \\ \text{if } U_{\text{T}} \neq 0 & \text{then } R \in \partial\mathbf{C}, \text{and it exists} \\ & a \geq 0 \text{ such that } R_{\text{T}} = -aU_{\text{T}} \end{cases} \tag{6}$$

where $\mathbf{C}$ is the Coulomb friction cone, $\mathbf{C} = \{R, | \ \|R_{\text{T}}\| \leq \mu R_{\text{N}}\}$ with μ the coefficient of friction. Coulomb's friction can also be formulated compactly as a complementarity condition on second–order cones as [3, 5, 4]

$$-\hat{U} \triangleq - \begin{bmatrix} U_{\text{N}} + \mu\|U_{\text{T}}\| \\ U_{\text{T}} \end{bmatrix} \text{ and } \mathbf{C}^{*} \in \hat{U} \perp R \in \mathbf{C} \tag{7}$$

where $\mathbf{C}^{*}$ is the dual cone of $\mathbf{C}$ [15].

1.2 Siconos

SICONOS is an Open Source scientific software primarily targeted at modeling and simulating nonsmooth dynamical systems in the most generic sense. In particular, it can deal with the following systems: mechanical systems (Rigid body or solid) with unilateral contact and Coulomb friction as we find in nonsmooth mechanics, contact dynamics or Granular material, switched Electrical Circuit such as electrical circuits with ideal and piecewise linear components Power converter, Rectifier, Phase-locked loop (PLL) or Analog-to-digital converter, sliding mode control systems. Other applications are found in Systems and Control (hybrid systems, differential inclusions, optimal control with state constraints), Optimization (Complementarity systems and Variational inequalities), Biology (Gene regulatory network), Fluid Mechanics and Computer graphics. The software is based on 4 main components :
1. SICONOS/NUMERICS (C API). Collection of low-level algorithms for solving basic Algebra and optimization problems arising in the simulation of nonsmooth dynamical systems: Linear complementarity problems (LCP) Mixed linear complementarity problems (MLCP) Nonlinear complementarity problems (NCP) Quadratic programming problems (QP) Friction-contact problems (2D or 3D) (Second-order cone programming (SOCP)) Primal or Dual Relay problems
2. SICONOS/KERNEL. C++ API that allows one to model and simulate nonsmooth dynamical systems. it contains : Dynamical systems classes (first order one, Lagrangian systems, Newton-Euler systems) and nonsmooth laws (complementarity, Relay, FrictionContact, impact)
3. SICONOS/MECHANICS This component is a modeling toolbox for multi-body systems based on external collision and geometry libraries.
4. SICONOS/FRONT-END generated python bindings for Numerics, Kernel and Mechanics, with a special support for Siconos data structures.

The developers team focused on the development of the simulation and modeling tools which are as reusable as possible for different scientific contexts.

Any specific pre/post-processing tools has been developed. Dynamical systems and interactions (unilateral constraints) classes provided by Siconos/Kernel may be configured with the help of plugins or by class derivation. For high–level applications SICONOS relies on external software and provides with an efficient computational engine. Concerning the mechanical modeling, it is possible to simulate linear or non-linear rigid or deformable bodies.

The numerical simulation strategies available in Siconos/Kernel are quasi-statics, smooth dynamics, nonsmooth dynamics, each one being either linear or non-linear and simulated with event–capturing or event–detecting schemes. Available time integration schemes are Moreau–Jean scheme, Schatzman–Paoli scheme, nonsmooth Newmark scheme [6, 7], Odepack suite and DAE solvers developed by E. Hairer [9].

Inside SICONOS/KERNEL, the description of the user model relies internally on a primary graph with dynamical systems as nodes and interactions as edges. During time evolution, the determination of active

constraints corresponds to the building of sub-graphs of the primary graph. In the case of problems formulated in local coordinates an adjoin graph transformation is dynamically applied to sub graphs in order to drive the computation of the components of the optimization problem given to Numerics solver.

The primary graph structure may be entirely specified once by the user, or it may also be rebuilt during the simulation, as, for example, the result of a contact detection broad-phase. Links with pre/post processing software for collision detection (Bullet contact detection [11]) and CAD libraries (OpenCascasde [16] and PythonOCC [13]) has been set–up. The latter link enables the contact detection between bodies defined by B-Rep (splines, nurbs, ...) and then to compute gaps and local frame at contact. Such a geometrical representation of the data allows to use realistic modeling of surfaces with large sliding. The possible impact laws for mechanics are complementary condition, impact with or without friction.

SICONOS is written in C++ and has an object oriented architecture, there is no graphical user interface (GUI). A simulation is run through a C++ or Python script in which are gathered the objects creations and the method calls on these object that make up the computation. Furthermore, not only the coarse level of command driving is available, but the time loop can be exploded in every step to allow for an interactive simulation. In the same way the whole database is accessible through copy or reference, allowing for a efficient access and aimed at designing any post-processing functions.

More details can be found at `http://siconos.gforge.inria.fr`.

1.3 Motivation for using the Nonsmooth Mechanics approach

Here are listed some of the main interests of using an approach based on nonsmooth Mechanics when we want to deal with multi–body systems with unilateral contacts, friction and impacts:
1. The use of complementarity formulation allows us to deal with hard contacts in a very efficient way avoiding the issues related to the time–integration of stiff problems. When we want to simulate contacts on hard surfaces, avoiding deep inter–penetrations obliges to use large stiffness parameters in a standard approach. This is overcame in the presented approach.
2. Nonsmooth contact laws are based on few parameters (coefficient of restitution and friction). Contact parameters are often very difficult to measure and have a great variability. Proposing contact laws with few parameters that represents the main effects at contact (inter–penetrability avoidance, friction with real sticking phase and plastic dissipation) provides the user with robust simulation tools in view of Control applications. Furthermore, nonsmooth contact laws can be easily extended to take into account more enhanced effects such as stiffness, damping or cohesion[3]. For soft–soil simulation, piecewise continuous Bekker model could be implemented in this context.
3. The associated simulation methods are proven to be efficient on a large number of applications from medium to large scale multi–body systems [14]. Granular materials with large number of contacts points (between 10^4 and 10^6) can be simulated with such algorithms.
4. Such an approach has been proved to be robust and efficient on several important industrial applications. Siconos is for instance used for more than 5 years in a strong collaboration between Schneider Electric and INRIA for the virtual prototyping of the mechanisms of circuit breakers [1]. The robustness with respect to contact parameters and the efficiency of the simulation that avoids problems due to stiffness provides Schneider Electric with a robust predictive tool for new designs. This numerical methods are also the object of a closed collaboration with ANSYS for the rigid body dynamics toolbox[8].

1.4 Rover simulation in Siconos

The Rover simulation in the Siconos platform allows us to simulate the robot motion taking into account rigorously dynamics and non-regular models of contact friction. Roughly, coding a Siconos application consists in two main steps : the first is to describe the dynamic Rover model and the contact model interaction between wheels and the environment; the second is to define appropriate numerical parameters for the model.

More precisely, any Siconos code (C++ or Python) is composed of the following sub-steps:

1. define some `DynamicalSystem` instances

2. define some `Interaction` instances, and for each `Interaction`:
   define `Relation`
   define `NonSmoothLaw`
3. build a `NonSmoothDynamicalSystem` : all `DynamicalSystem` + all `Interaction` objects arranged into a directed `Graph`
4. build a `Model` that holds the `NonSmoothDynamicalSystem`.
5. define a `Simulation`: the way the behavior of the `NonSmoothDynamicalSystem` will be computed:
   define a `TimeDiscretisation`
   choose a strategy: `TimeStepping` or `EventDriven`
   define some integrators for the DynamicalSystems (`OneStepIntegrator`)
6. define a way to formalize and solve the possible non smooth problems (formulation + solver, the `OneStepNSProblem`)
7. associate the `Simulation` to the Model
8. launch the `Simulation` (time loop)

1.5 Modeling of the system

The Rover dynamical model is a LagrangianDS siconos object (lagrangian formulation). For 3D Rover modeling, the inertia matrix and contact model interaction relations are difficult to set-up by hands. We use a Maple™ code to automatically generate these relations, and to export into C files. In Siconos, we import these relations as a plug-in function that are dynamically loaded. The HuMAns toolbox([17]) using Maple™ software tool enable us to easily do complex modeling easily with high accuracy.

Originally developed for humanoid robotics research, the HuMAnS toolbox (for Humanoid Motion Analysis and Simulation) proposes a wealth of state - of - the - art algorithms from the field of robotics research for the modeling, the analysis and the simulation. In our case, we used it to generate the multi-body dynamic model for the Rover. For this, the user need to describe the Rover parameters through several Maple™ input files. One file must describe kinematic parameters of the Rover, it indicates the relative location of key-points of the model and the degree of freedom of every connecting point. An other file describes the dynamical parameters, it describes the inertials parameters: the gravity vector, the mass of the segment, the position of the segments centers of mass and the inertia matrix of these segments relative to the centers of their attached frames.

Then, the Maple™ generator included in HuMAns produces C code for $M(q)$ the system inertia, $G(q)$ the forces which derived from a potential energy (for example the gravity), $N(q, \dot{q})$ a non-linear effects matrix.

For the simulation we used a simplified model inspired from the ExoMars rover chassis with six wheels on three simple passive bogies located forward at each side of the rover and one on the rear (see [12]). The payload is simply modelised with a mass uniform parallelepiped. The visual representation and joints notation are given on the figure 2. The rover has 21 Degree Of Freeedom (DOF) : 6 for the location of the rover, 6 for the rotation of each wheel, 6 for each wheel steering and 3 for each passive joint of the bogies.

To describe the contact of the Rover with the solid ground or granular soil in Siconos, we need two plugin functions: h to compute the distance between contact points and G which is the transformation matrix of velocity of contact points from the global frame to the local frame.

To simplify the model, we only consider point contact. In this contact model, we simplify the axis of the wheel into a point, and create contact model between this point and the sphere, a plane or a mesh.

For the simulation results given below, the ground is modeled by meshes. In order to be able to perform a simulation of the rover roaming on a triangular mesh in Siconos one needs to have the appropriate interaction and relation classes defined. Relation class is an interface providing distances between the surfaces being checked for collision as well as jacobians, which in turn provide mappings from the local to global coordinate frames. While distances are needed for collision detection, jacobians are crucial for resolving impacts and nonsmooth friction problems. Distances calculation is a broad topic in itself, briefly speaking these can be delivered by a third-party collision detection library or calculated according to an explicit algorithm built-in in the respective relation class. In the standalone simulation the latter approach has been chosen. Namely, distance function has been defined explicitly in order to calculate

the gap between a contact point on each wheel of the robot and a triangle arbitrarily oriented in space. Everytime this function returns zero (with certain tolerance) a collision is declared. Jacobian is defined within the relation class as an operator resulting from the multiplication of the local triangle's rotation matrix with an inverse of robot's geometric jacobian. Relation object together with a non-smooth law define the interaction between systems. In the standalone simulation an interaction is instantiated between each wheel and each triangle. Thus, there is six times as many interactions as wheels. This approach can become inefficient as the number of triangles (and complexity of the mesh) grow. To compensate for this problem a space filter should be used in the future which in essence filters out and discards interactions between objects which are far enough.

The C code generated, for the rover model and the contact model interaction, is then imported in Siconos code plugin function. It enables us to easily do complex modeling with high accuracy. This work of modeling is detailed in the report [10].

2 Mechanical tests of the rover

2.1 Parameters used in simulations

For below presented simulations following parameters have been manipulated:

- Vector of initial positions and velocities of the joints and the base q_0 and v_0
- Friction and normal restitution coefficient μ and eps_n
- Simulation time and timestep T and h
- Torques applied to the joints τ

For all simulations gravity coefficient on Mars have been used ($3.7\frac{m}{s^2}$).

2.2 Tests scenarios specification

Seven tests have been defined and performed:

1. Rover stabilization on a horizontal plane after a free-fall phase.

   In the first test setting the rover falls freely from the height of 2 meters onto a horizontal plane. No torques are applied in any of the joints. The only external forces acting on the rover are the gravity and the ground reaction forces. Initial position of the rover has been set to (x, y, z) = (5, 5, 2) [m]. Friction coefficient has been set to 0.7. This case has been divided into two sub-cases: first one with the normal restituion coefficient set to zero and the second one with the normal restitution coefficient set to 0.2. Tangential restitution coefficients have been set to zero in both sub-cases.

2. Rover stabilization on a horizontal plane with linearly varying velocity of one steering axis.

   In the second test setting rover is dropped onto a horizontal plane and stands idle on it during the first 50 seconds of the simulation. After 50 seconds from the beginning of the simulation a constant torque $\tau = 0.00002Nm$ is applied in one of the steering axes (steering axis FL) causing its rotational motion with linear velocity. Wheel makes two full rotations around its vertical steering axis (FL). Other external forces acting on the rover are the gravity and ground reaction. Initial position of the center of mass of the robot has been set to (x, y, z) = (5, 5, 2) [m]. Friction coefficient has been set to 0.4. Restitution coefficients (tangential and normal) have been set to zero.

3. Rover stabilization on a horizontal plane with acceleration and deceleration phases.

   In the third test setting rover's parcour is divided into six time intervals with different values of torques applied to the wheels:

   - $0s < t_s < 20s, \tau = 0Nm$

- $20s < t_s < 40s$, $\tau = 0.0007Nm$
- $40s < t_s < 80s$, $\tau = 0Nm$
- $80s < t_s < 105s$, $\tau = -0.0007Nm$
- $105s < t_s < 150s$, $\tau = 0Nm$
- $150s < t_s < 200s$, $\tau = -0.0005Nm$

Other external forces acting on the rover are the gravity and ground reactions. Initial position of the center of mass of the robot has been set to (x, y, z) = (5, 8, 0.57) [m]. In the 5^{th} phase rover effectively stops its motion until negative torques are applied. Friction coefficient has been set to 0.7. Restitution coefficients (tangential and normal) have been set to zero.

4. Rover stabilization on an inclined plane with a phase of upward motion.

   In the fourth test setting rover is set to stand steadily on an inclined plane. Inclination angle of the slope has been set to 10°. Torques applied to the wheels counterbalance torques caused by the gravity. After initial period higher torques are applied to the wheels which cause the robot to drive upwards. Blocking torques are equal to $\tau_b = -0.87072Nm$. During the second phase of motion linearly varying torques are applied to all wheels.

   Rover's parcour has been devided into two phases:

   - $0s < t_s < 30s$, $\tau_b - constant\ blocking\ torques$
   - $30s < t_s < 40s$, $\tau_m - linearly\ increasing\ torques$

   Friction coefficient has been set to 0.8. Restitution coefficients (tangential and normal) have been set to zero.

5. Rover stabilization on an inclined plane with acceleration and deceleration phases.

   In the fifth test setting rover is dropped onto the inclined plane. Inclination angle of the slope has been set to 10°. After the 10^{th} second of motion a simple PD controller is set using position and velocity of the center of mass in order to stop the rover on the slope. Rover stops effectively at $t = 25s$. Friction coefficient has been set to 0.8. Restitution coefficients (tangential and normal) have been set to zero.

6. Spherical obstacle crossing on a horizontal plane.

   In the sixth test setting rover is dropped onto the horizontal plane from the height of $2\ m$. After $10\ s$ linearly varying torques are applied to all wheels. A spherical obstacle has been set in front of the rover. The obstacle is in the form of a sphere which protudes outside the plane. Center of the sphere has been set so that the protrusion is equal to $20\ cm$. Coefficients of friction and restitution (normal) are in this case respectively: 0.3 and 0.

7. Vertical step obstacle crossing.

   In the seveth test setting rover is dropped onto a horizontal plane. An obstacle in the form of a horizontal step of height $0.1m$ (roughly equal to the wheel radius) has been set in front of the rover. At certain point of time ($t = 10s$) rover starts moving towards the step and crosses it mounting on the higher plane. Friction coefficient has been set to 0.6. Restitution coefficient (normal) has been set to 0.1.

Results of each of the above tests will be detailed in the following sections.

In mechanical simulations with nonsmooth approach one is mostly interested in the following quantities:

- x_{COM} - mass center coordinates
- x_{wheels} - wheels angular displacement
- v_{COM} - mass center velocity

- v_{wheels} - wheels angular velocity
- R - reaction forces (impulsions) in lagrangian (global) coordinates
- λ_N ($\lambda_{\bar{n}}$) - normal component of the contact force (impulsion) in local coordinates
- λ_{T_x} ($\lambda_{\bar{t}}$) - tangential component of the contact force (impulsion) in local coordinates in the x direction
- λ_{T_z} ($\lambda_{\bar{s}}$) - tangential component of the contact force (impulsion) in local coordinates in the z direction
- y_N ($y_{\bar{n}}$) - gap function (distance between contact point and the constraint function)
- $\dot{y}_N$ ($\dot{y}_{\bar{n}}$) - normal component of the local contact velocity
- $\dot{y}_{T_x}$ ($\dot{y}_{\bar{t}}$) - tangential component x of the local contact velocity
- $\dot{y}_{T_z}$ ($\dot{y}_{\bar{s}}$) - tangential component z of the local contact velocity

For each scenario a subset of the above quantities has been plotted.

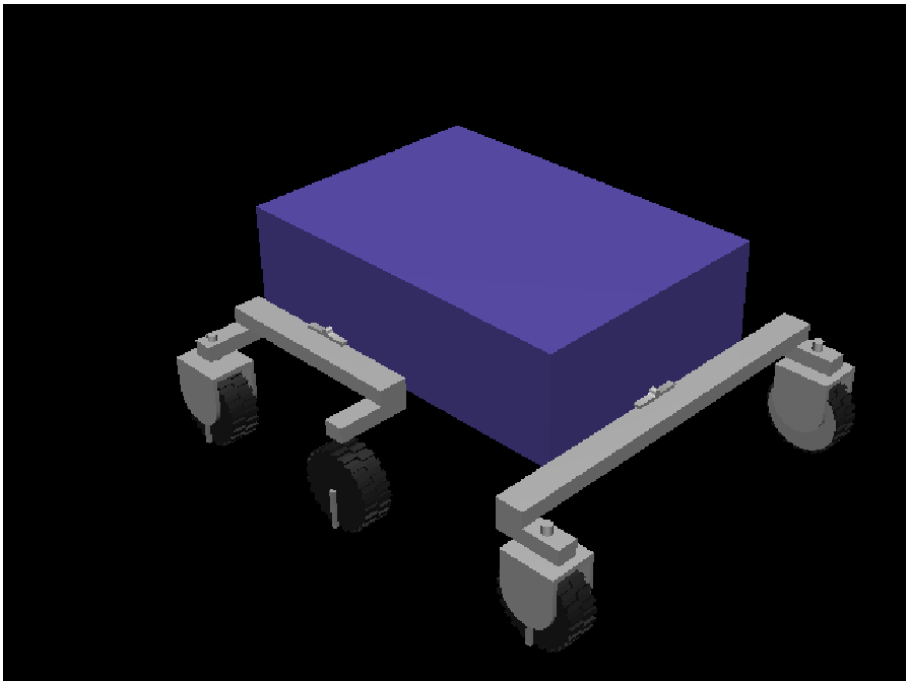


Figure 1: Assumed model of the robot

Simplified kinematic structure of the robot is as follows:

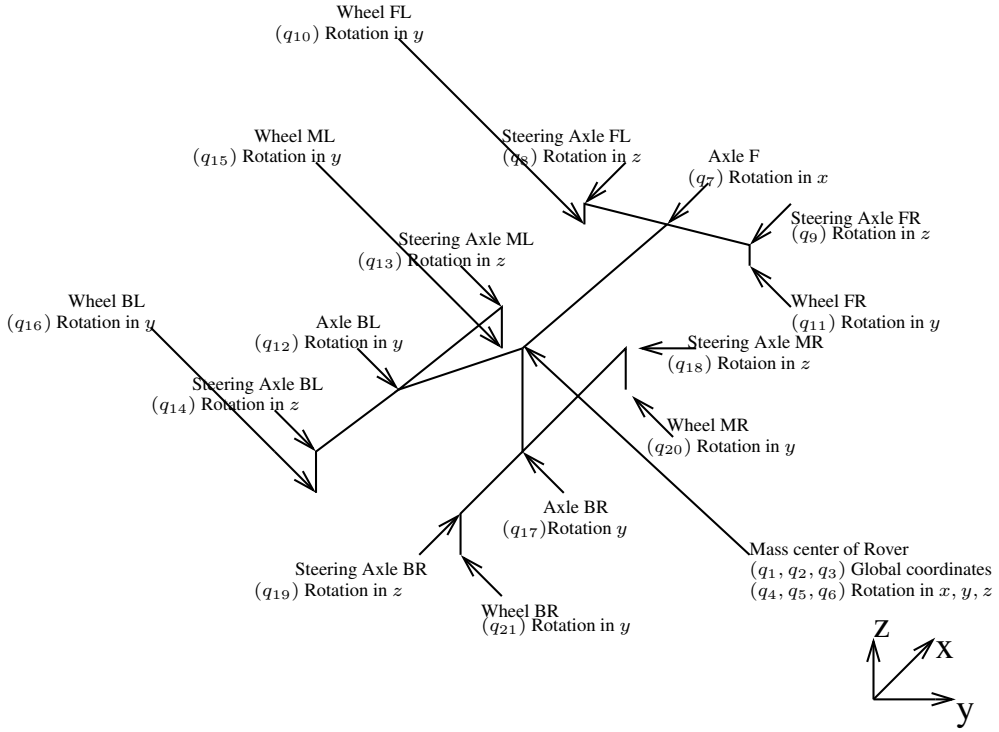


Figure 2: Geometry of the model

3 Test 1 - Rover stabilization on a horizontal plane

This test is divided into two sub-cases.

3.1 Normal restitution coefficient set to $eps_n = 0.0$

In the first test setting rover falls freely from the height of 2 meters onto a horizontal plane. No torques are applied in any of the joints. The only external forces acting on the rover are the gravity and the ground reaction forces. Initial position of the rover has been set to (x, y, z) = (5, 5, 2) [m]. Friction coefficient has been set to 0.7. In this sub-case restitution coefficients (tangential and normal) have been set to zero.

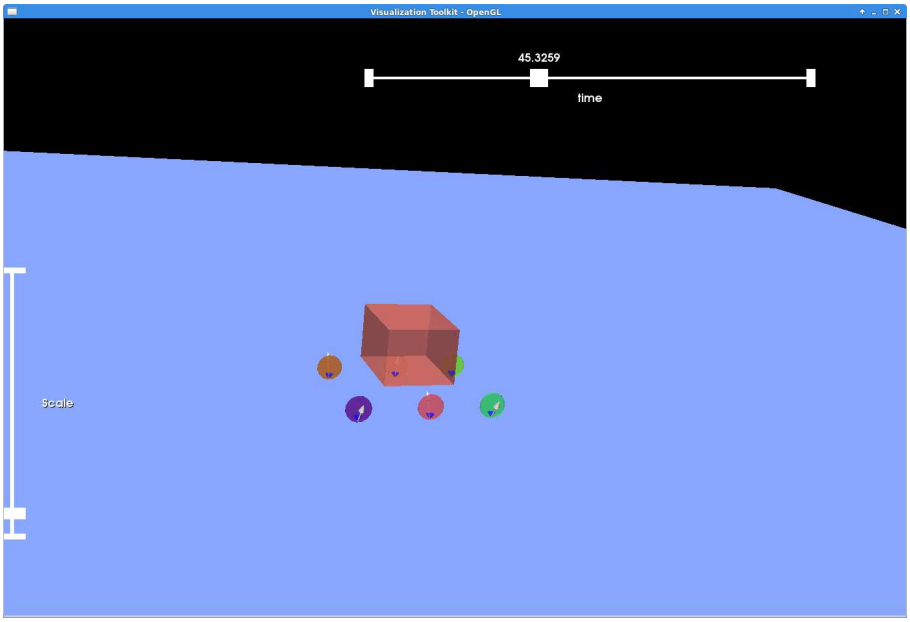


Figure 3: First test setting (arrows depict contact forces)

The following essential quantities have been plotted in this case:

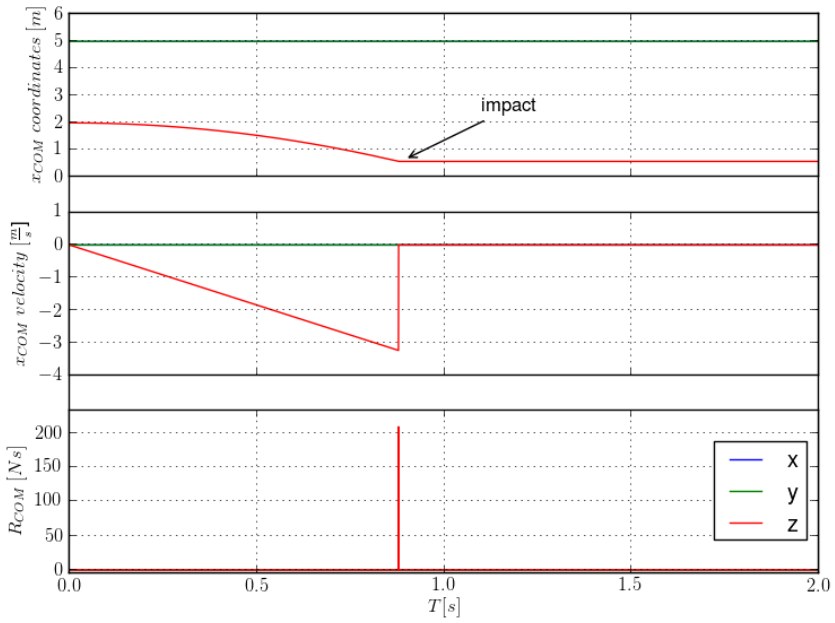


Figure 4: Position, velocity and reaction forces of the center of mass

Comments

As seen in the figure 4, coordinate z of the center of mass of the rover decreases parabolically until the system is stable on the plane. This represents the free-fall phase under gravity. After reaching the plane there is no rebound as the coefficient of restitution in the nonsmooth contact law has been set to zero. Impact corresponds to a sharp peak in the force as seen in the third sub-figure.

Also, the following complimentary quantities have been plotted:

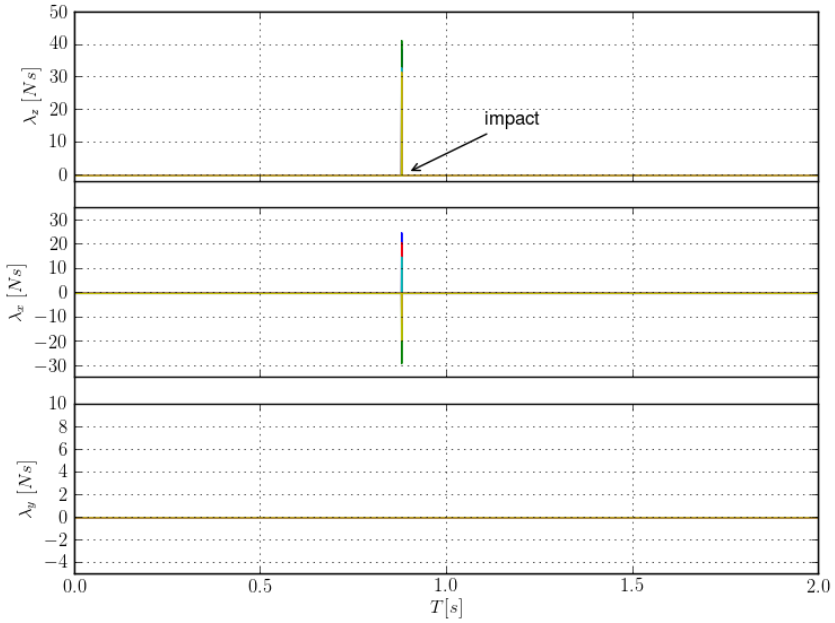


Figure 5: Normal and tangential components of the local contact force (impulsion λ) for each wheel during impact

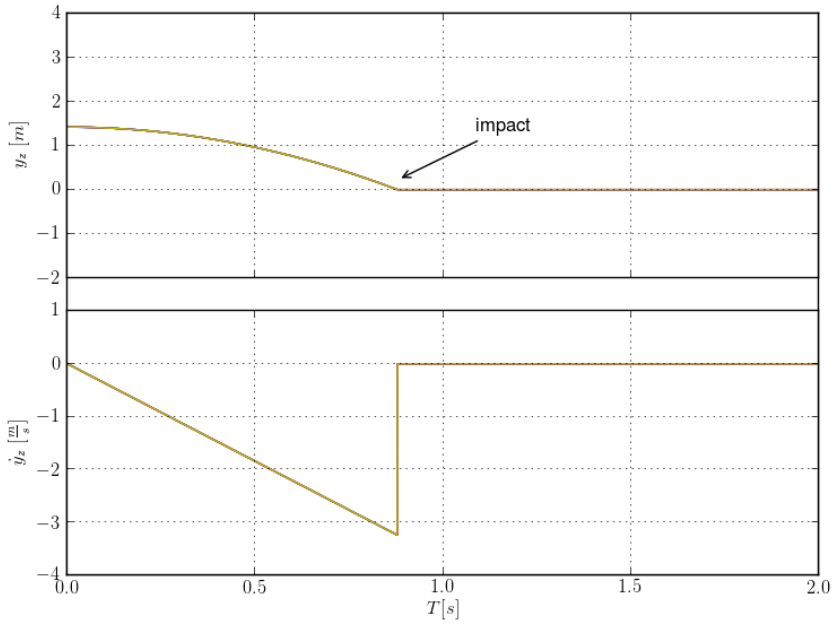


Figure 6: y_N - gap function (distance between contact point and the constraint function) for each wheel and $\dot{y}_N$ - normal component of the local contact velocity for each wheel

Comments

Plots in the figure 5 correspond to forces (normal and tangential impulsions) computed in the local contact frame. Contact forces are always computed in the local frame and then expressed in the coordinates of the system using jacobian matrix of the constraint function. Constraint function (gap function) used to detect the contact and the relative velocity are plotted in the figures 6. Gap function is a relative distance between two contacting bodies and it imposes geometric constraints on the system. Once slightly below zero it will trigger a nonsmooth interaction between those bodies.

3.2 Normal restitution coefficient set to $eps_n = 0.2$

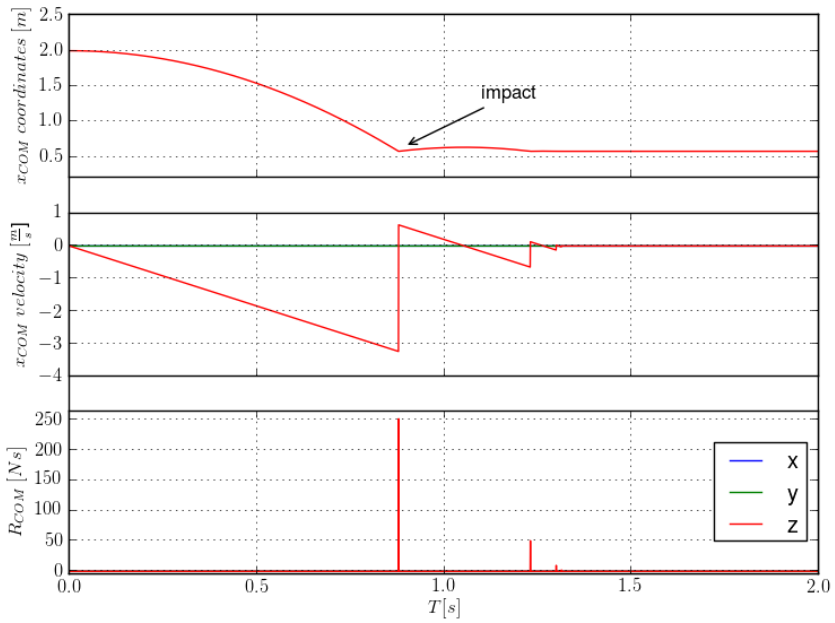


Figure 7: Position, velocity and reaction forces of the center of mass

Comments

As the restitution coefficient has been set to a non-zero value in this case one observes multiple impacts. Impacts corresponds to peaks of forces in the third sub-figure and nonsmooth changes of velocity in the second sub-figure. As the energy is dissipated the magnitudes of impacts reduce in time.

Also, the following complimentary quantities have been plotted:

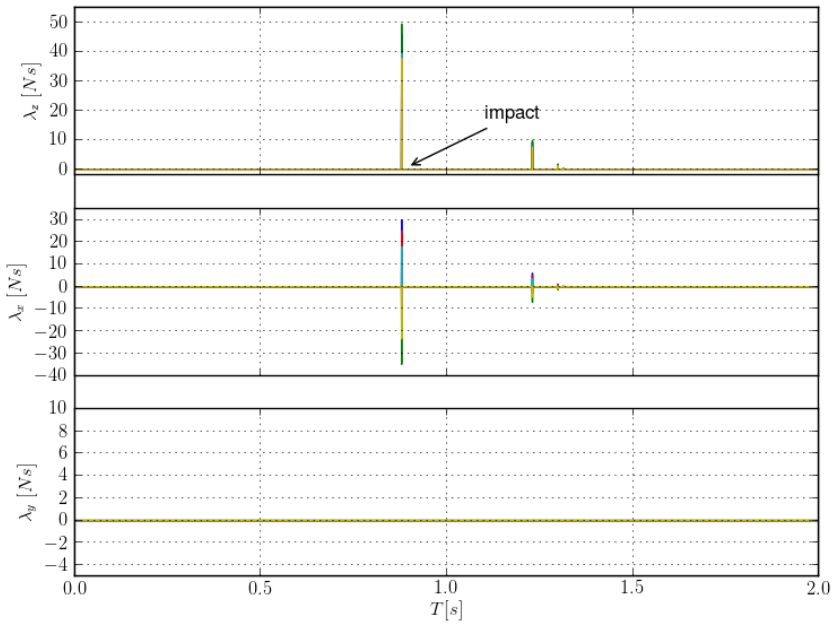


Figure 8: Normal and tangential components of the local contact force (impulsion λ) for each wheel during impact

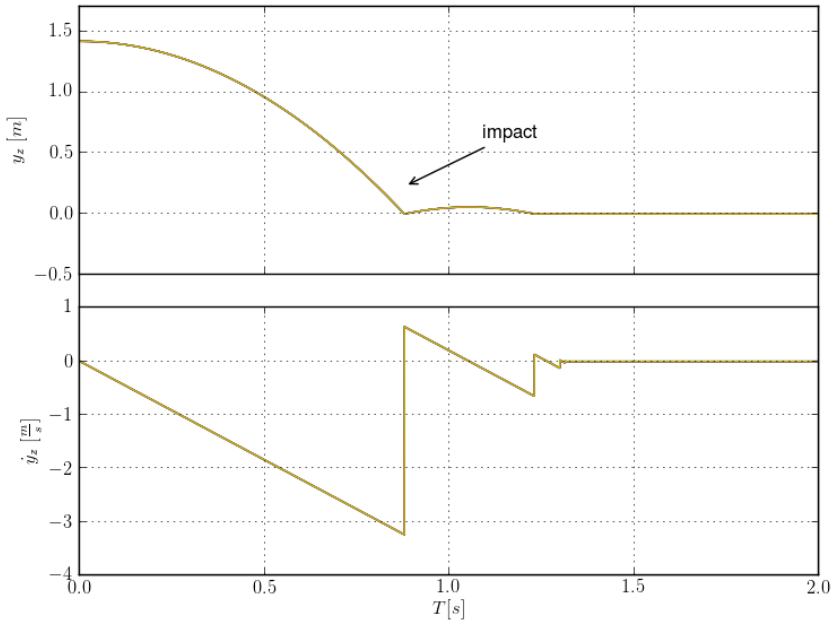


Figure 9: y_N - gap function (distance between contact point and the constraint function) for each wheel and $\dot{y}_N$ - normal component of the local contact velocity for each wheel

Comments

In the figure 9 one can see the gap function and relative velocity plots. With respect to the previous sub-case rebounds off the ground are visible.

4 Test 2 - Rover stabilization on a horizontal plane with varying velocity of one steering axis

In the second test setting rover is dropped onto a horizontal plane and stands idle on it during the first 50 seconds of the simulation. After 50 seconds from the beginning of the simulation a constant torque $\tau = 0.00002Nm$ is applied in one of the steering axes (steering axis FL) causing its rotational motion with linear velocity. Wheel makes two full rotations around its vertical steering axis (FL). Other external forces acting on the rover are the gravity and ground reaction. Initial position of the center of mass of the robot has been set to (x, y, z) = (5, 5, 2) [m]. Friction coefficient has been set to 0.4. Restitution coefficients (tangential and normal) have been set to zero.

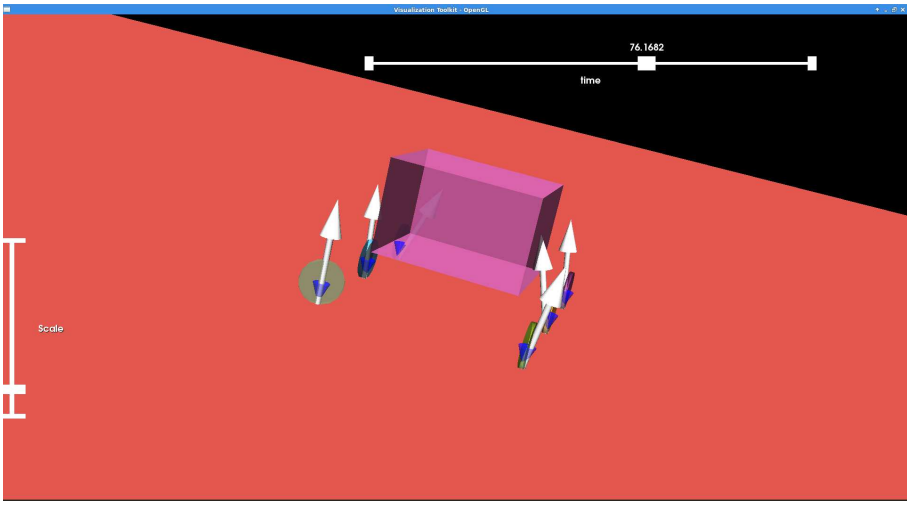


Figure 10: Second test scenario (arrows depict contact forces)

In this case, following essential quantities have been plotted:

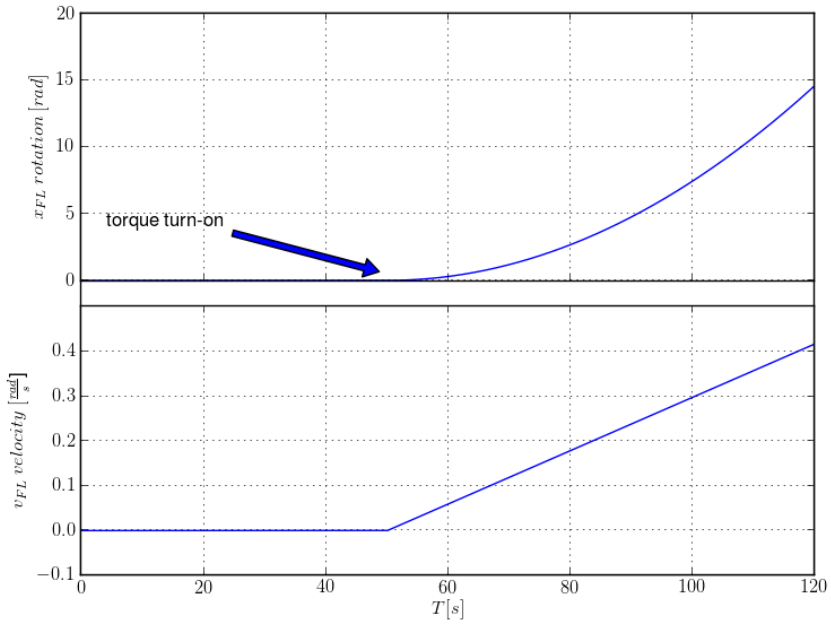


Figure 11: x_{FL}, v_{FL} - angular displacement and velocity of the FL axis

Comments

In the figure 11 one can see the variation of the steering axis displacement and velocity. After a constant torque is applied the axis is rotating with a linear velocity which corresponds to the quadratic shape of the displacement curve.

Also, the following complimentary quantities have been plotted:

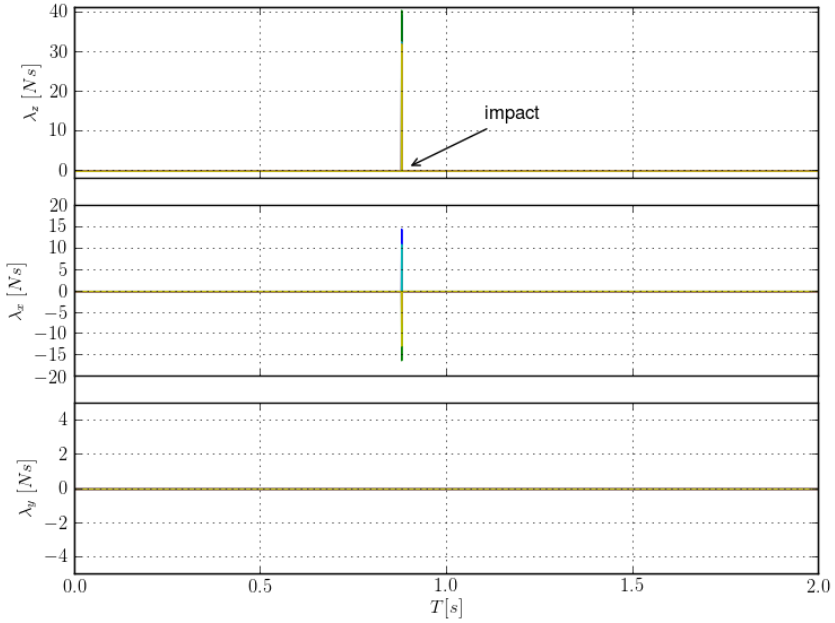


Figure 12: λ_N - normal component of the contact force (impulsion) for each wheel

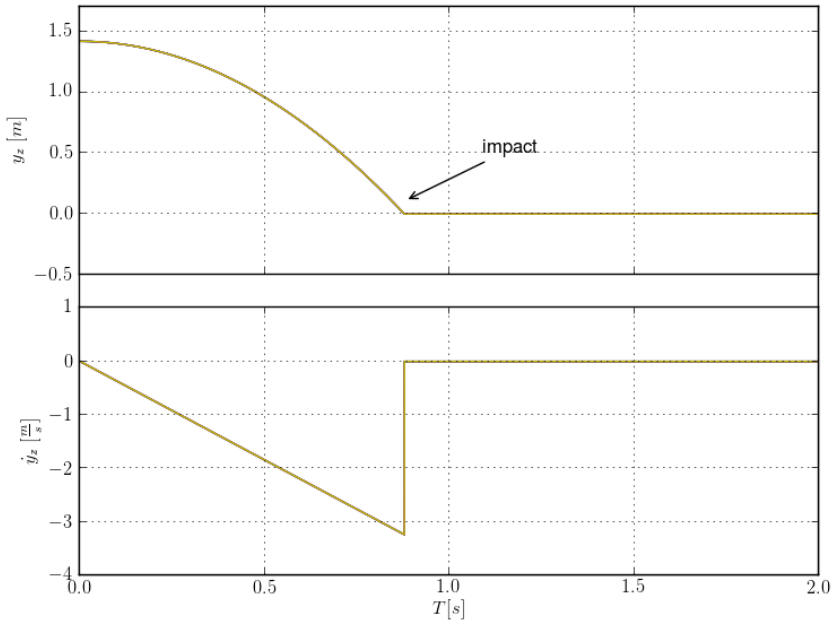


Figure 13: y_N - gap function (distance between contact point and the constraint function) for each wheel

Comments

In the figure 12 one can observe one impulsion (force) peak corresponding to the impact when the rover falls onto the ground. In the figure 13 the gap function and relative velocity are displayed.

5 Test 3 - Rover stabilization on a horizontal plane with acceleration and deceleration phases

In the third test setting rover's parcour is divided into six time intervals with different values of torques applied to the wheels:

1. $0s < t_s < 20s, \tau = 0Nm$
2. $20s < t_s < 40s, \tau = 0.0007Nm$
3. $40s < t_s < 80s, \tau = 0Nm$
4. $80s < t_s < 105s, \tau = -0.0007Nm$
5. $105s < t_s < 150s, \tau = 0Nm$
6. $150s < t_s < 200s, \tau = -0.0005Nm$

Other external forces acting on the rover are the gravity and ground reactions. Initial position of the center of mass of the robot has been set to (x, y, z) = (5, 8, 0.57) [m]. In the 5^{th} phase rover effectively stops its motion until negative torques are applied. Friction coefficient has been set to 0.7. Restitution coefficients (tangential and normal) have been set to zero.

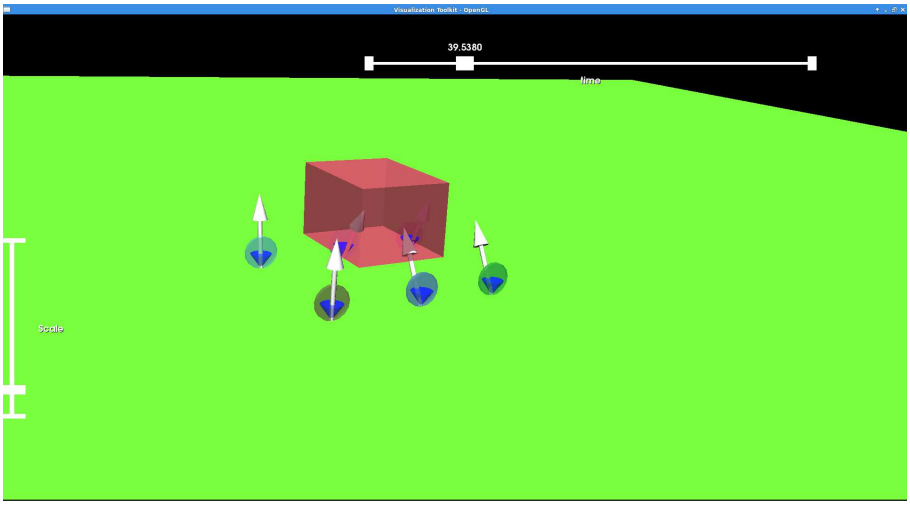


Figure 14: Third test scenario (arrows depict contact forces)

In this case, following quantities have been plotted:

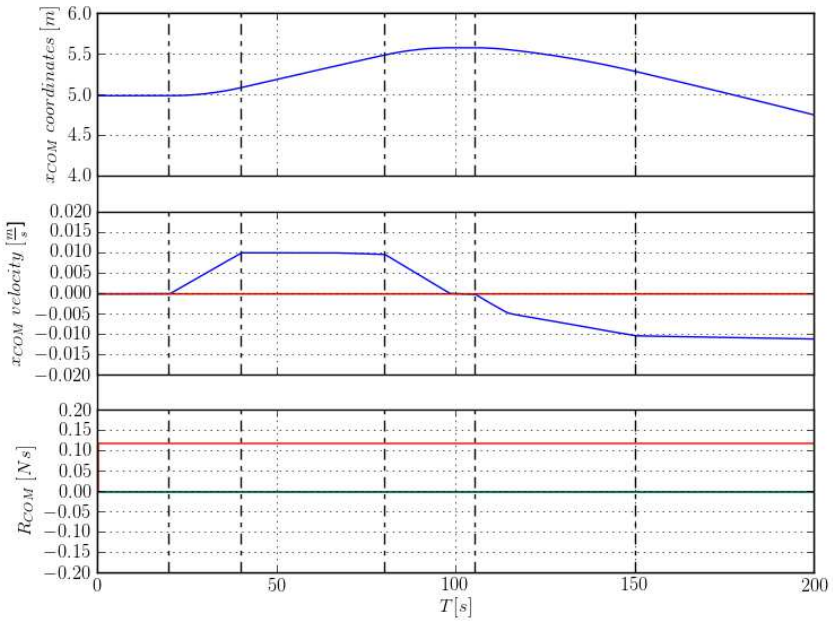


Figure 15: Position, velocity and reaction forces of the center of mass

Comments

Rover's motion has been divided into six phases. Figure 15 depicts those phases in terms of the center of mass velocity. Phases 2 and 3 represent forward motion with a constant velocity in the third phase. In the fourth phase rover moves in the opposite direction with a linear velocity to effectively stop in the fifth phase of motion.

Following additional quantities have also been plotted:

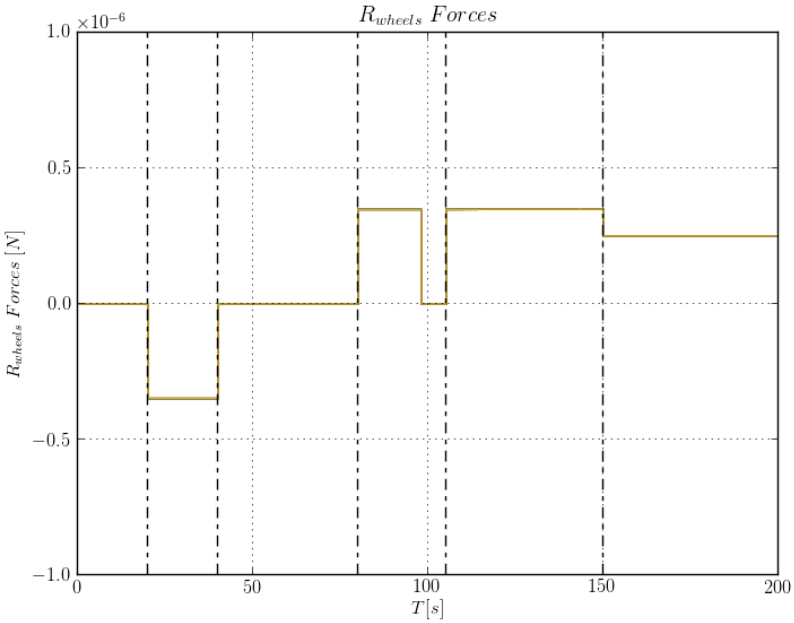


Figure 16: R_{wheels} - reaction forces (impulsions) of wheels in lagrangian coordinates

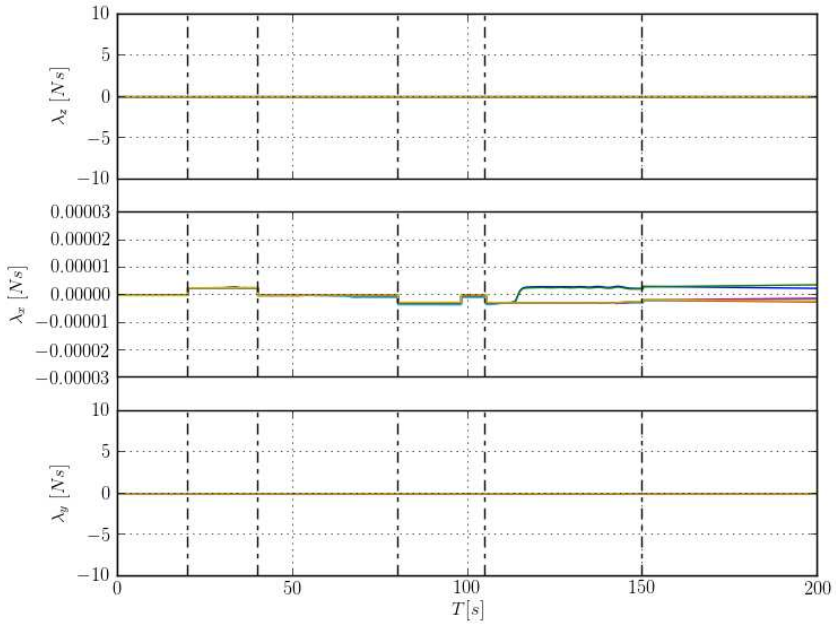


Figure 17: λ_N, λ_{T_x}, λ_{T_y} - normal and tangential components of the contact force (impulsion) for each wheel

Comments

In the figure 16 one can see the step pattern of the reaction torques resulting from the fact that the rover is moving along the x axis. Jumps in value represent changes in the friction force along x axis as torques are applied to the wheels. In the figure 17 all components of the local contact forces are displayed. Step pattern along x axis can be observed.

6 Test 4 - Rover stabilization on an inclined plane with a phase of upward motion

In the fourth test setting rover is set to stand steadily on an inclined plane. Inclination angle of the slope has been set to 10°. Torques applied to the wheels counterbalance torques caused by the gravity. After initial period higher torques are applied to the wheels which cause the robot to drive upwards. Blocking torques are equal to $\tau_b = -0.87072Nm$. During the second phase of motion linearly varying torques are applied to all wheels.

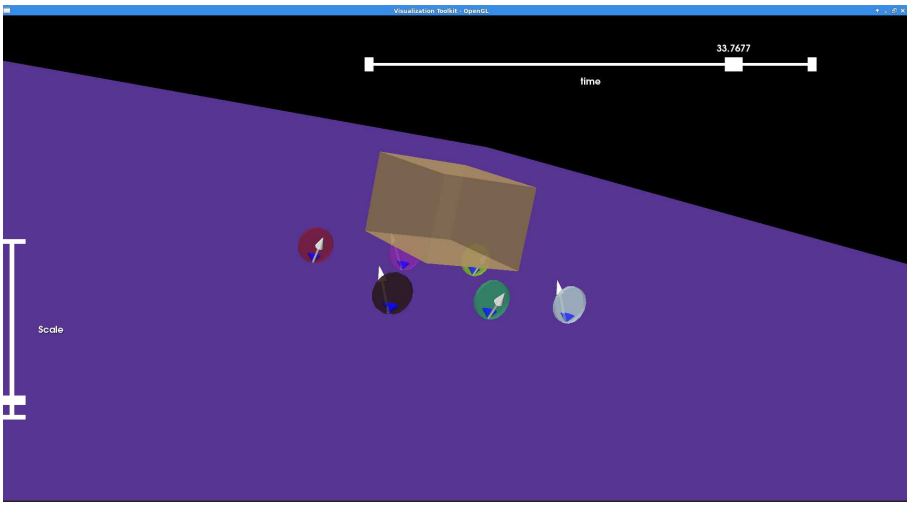


Figure 18: Fourth test scenario

Rover's parcour has been devided into two phases:

1. $0s < t_s < 30s, \tau_b - constant\ blocking\ torques$
2. $30s < t_s < 40s, \tau_m - linearly\ increasing\ torques$

Friction coefficient has been set to 0.8. Restitution coefficients (tangential and normal) have been set to zero.

In this case, following essential quantities have been plotted:

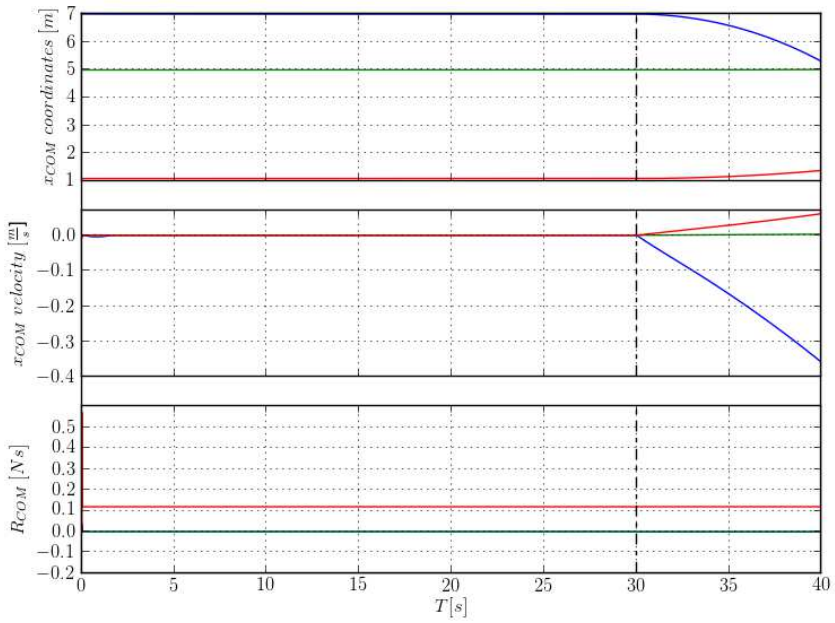


Figure 19: x_{COM} - mass center coordinates

Comments

Two phases are distinguished in this case. In the figure 19 one can see that in the first phase rover is motionless because of the constant blocking torques applied to all wheels. These torques through friction nullify the effect of gravity on the system. Coordinates of the center of mass are constant until the second phase. This means that the friction-contact model is efficient in the sense that it matches closely physical behavior of the phenomenon. One can see this more clearly in the velocity curve where the velocity of the center of mass is zero in the first phase of motion. Nonsmooth friction-contact model allows for this effect as it does not rely on regularization of friction laws. Last curve in the figure 19 represents the ground reaction forces applied to the mass center of the system. This force increases very slightly as the rover starts moving upwards because of the friction force.

Following additional quantities have been plotted:

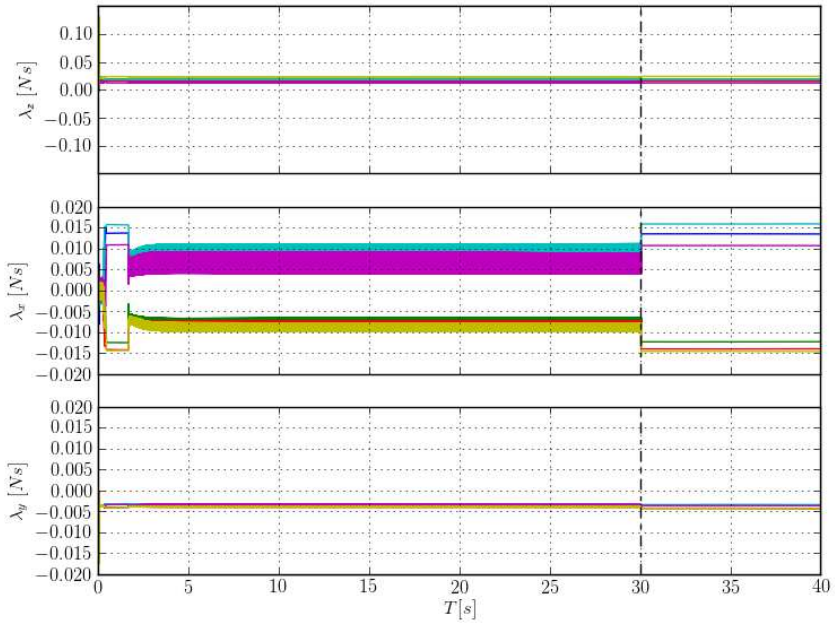


Figure 20: λ_N, λ_{T_x}, λ_{T_z} - normal and tangential components of the contact force (impulsion) for each wheel

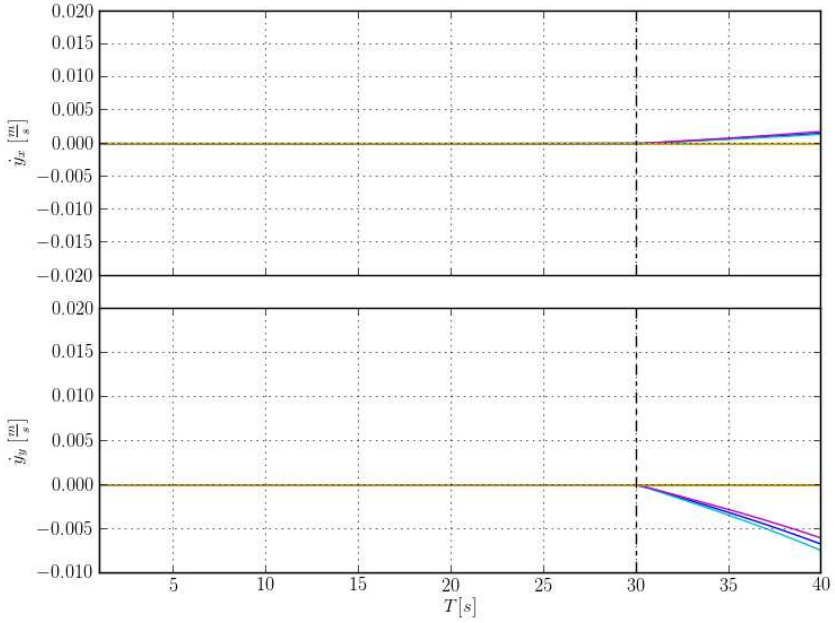


Figure 21: $\dot{y}_{T_x}$, $\dot{y}_{T_y}$ - tangential components of the local relative velocity for each wheel

Comments

In the figure 20 one can see normal and tangential components of the local contact forces. Immediate observation is that the projections of forces oscillate during the first phase of motion. This is due to the fact that contact solver finds a different solution to the friction-contact problem at each time instant.

7 Test 5 - Rover stabilization on an inclined plane with acceleration and deceleration phases

In the fifth test setting the rover is posed on and inclined plane. Inclination angle of the slope has been set to 10°. After the 10^{th} second of simulation a simple PD controller is set using position and velocity of the center of mass, in order to stop the rover on the slope. Rover stops effectively at $t = 25s$. Friction coefficient has been set to 0.8. Restitution coefficients (tangential and normal) have been set to zero.

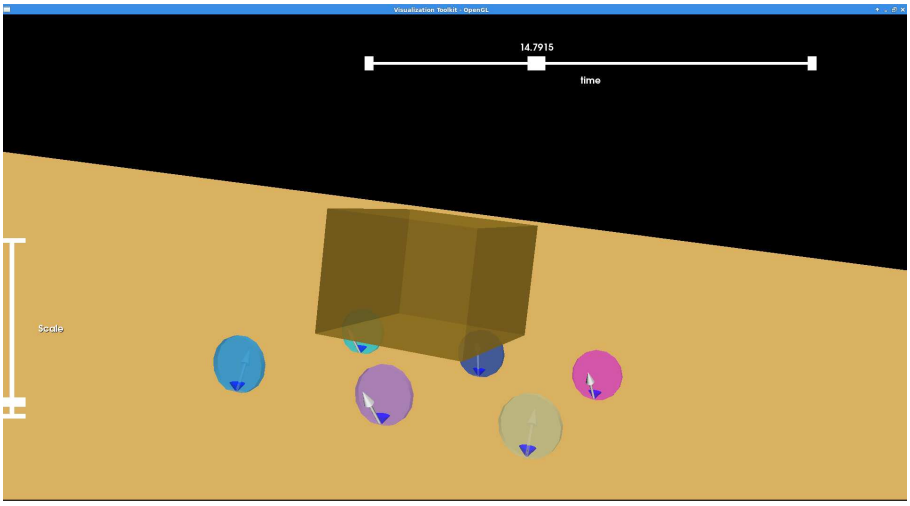


Figure 22: Fifth test scenario

In this setting we are mostly interested in two phases:

1. $0s < t_s < 10s$
2. $10s < t_s < 40s$

In this case, following essential quantities have been plotted:

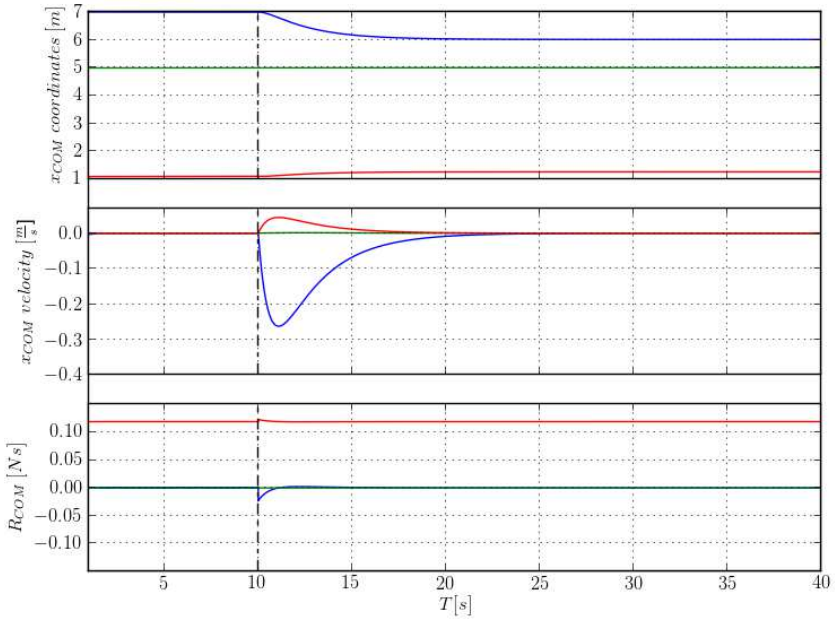


Figure 23: position, velocity and reaction force of the mass center

Comments

One can clearly see two phases of motion of the rover. In the end of the second phase rover stops its motion on the inclined plane which once again demonstrates the efficiency of the nonsmooth friction-contact law. This can be seen in the figure 23 where in the second phase velocity components converge towards zero. This effect corresponding to a true nature of friction and can be captured in numerical experiments using nonsmooth laws. Using regularized laws to describe friction would not allow the system to retain non-zero torques with zero velocity.

Following additional quantities have been plotted:

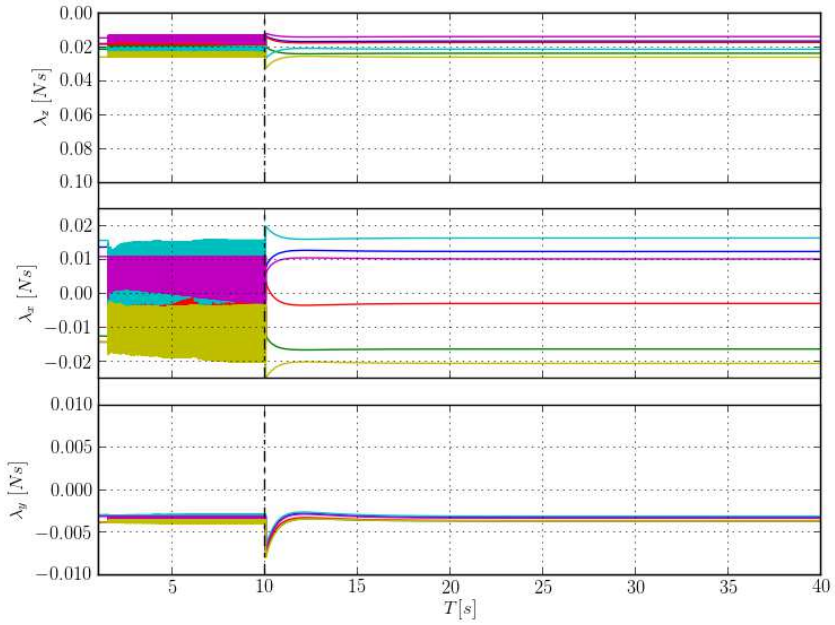


Figure 24: λ_N - normal component of the contact force (impulsion) for each wheel

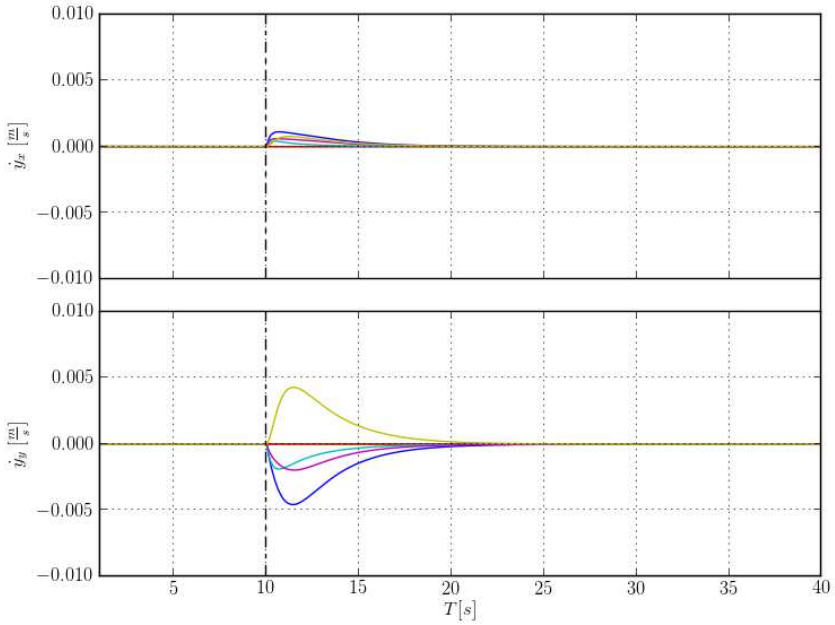


Figure 25: $\dot{y}_{T_z}$ - tangential component z of the local contact velocity for each wheel

Comments

As in the previous test setting, an immediate observation comes into mind as to the oscillations in the figure 24. Indeed, it is the same reason as previously that they appear in the plot.

8 Test 6 - Spherical obstacle crossing on a horizontal plane

In the sixth test setting rover is dropped onto the horizontal plane from the height of 2 m. After 10 s linearly varying torques are applied to all wheels. A spherical obstacle has been set in front of the rover. The obstacle is in the form of a sphere which protudes outside the plane. Center of the sphere has been set so that the protrusion is equal to 20 cm. Coefficient of friction has been set to 0.3 whereas the coefficient of restitution to 0.

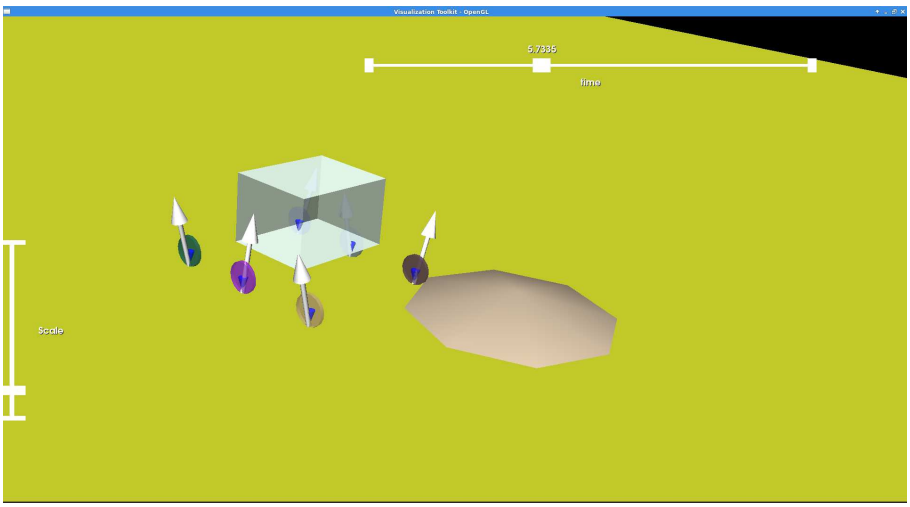


Figure 26: Sixth test scenario

In this case, the following essential quantities have been plotted:

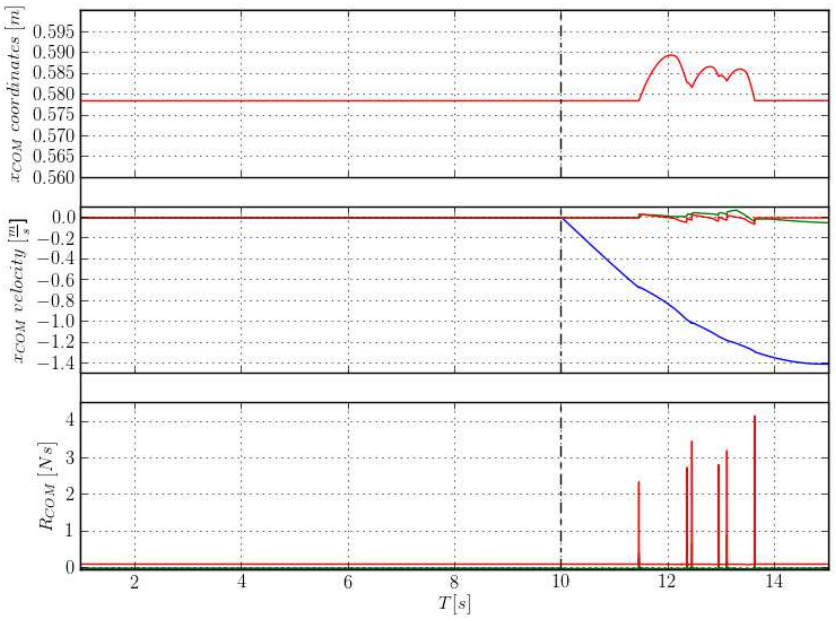


Figure 27: position, velocity and reaction forces of the mass center of the rover

Comments

In this setting attention can be drawn to the spherical obstacle crossing. In the figure 27 in the first sub-plot on can see z coordinate evolution of the mass center of the robot. It is clearly seen how around 11^{th} minute rover mounts the sphere. In the sub-plot below one can see the evolution of velocities of the mass center where several small jumps occur in the y and z dimension as a result of impacts. Velocity in the x direction is clearly larger as the rover moves along the x axis. Impacts are more clearly seen in the third sub-plot with each peak corresponding to an impact.

Following additional quantities have also been plotted:

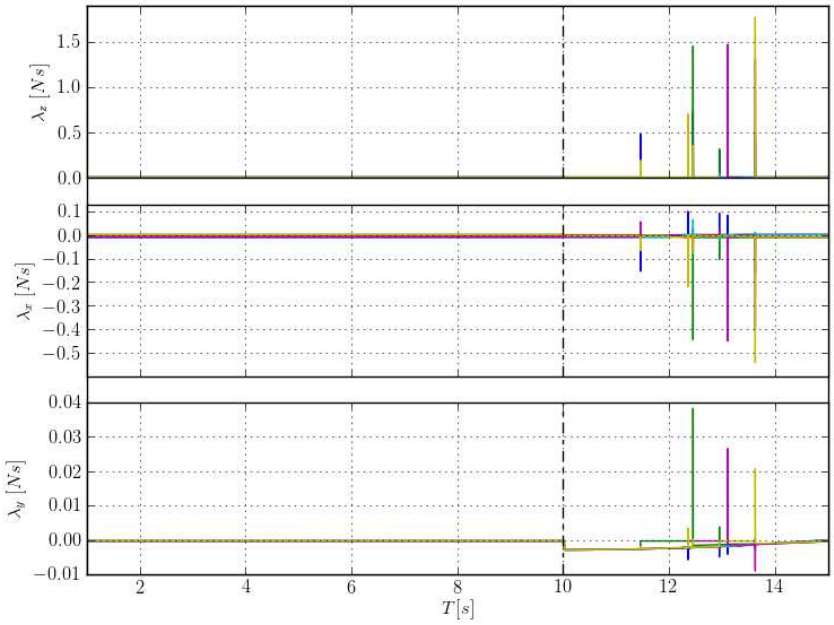


Figure 28: λ_N, λ_{T_x}, λ_{T_y} - normal component of the contact force (impulsion) for each wheel (interaction with the plane)

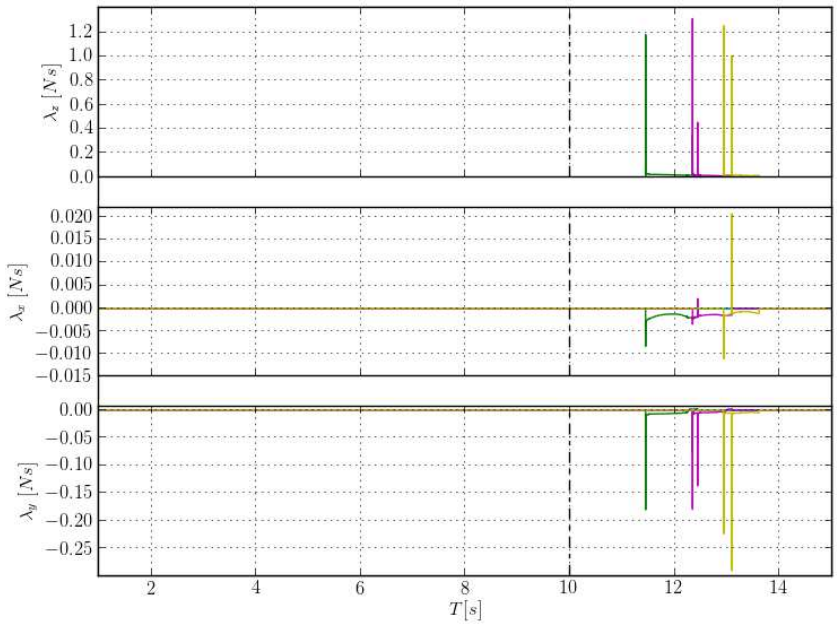


Figure 29: λ_N, λ_{T_x}, λ_{T_y} - normal component of the contact force (impulsion) for each wheel (interaction with the sphere)

Comments

In the figure 28 and 29 local contact forces have been plotted seen from the interaction with plane and sphere respectively. One can observe several peaks corresponding to the phase of the simulation when the obstacle is being crossed.

9 Test 7 - Veritcal step obstacle crossing

In the seventh test setting rover is dropped onto a horizontal plane. An obstacle in the form of a vertical step of height $0.1m$ has been set in front of the rover. At a certain point of time ($t = 10s$) rover starts moving towards the step and crosses it mounting on the higher plane. Friction coefficient has been set to 0.6. Restitution coefficient (normal) has been set to 0.1.

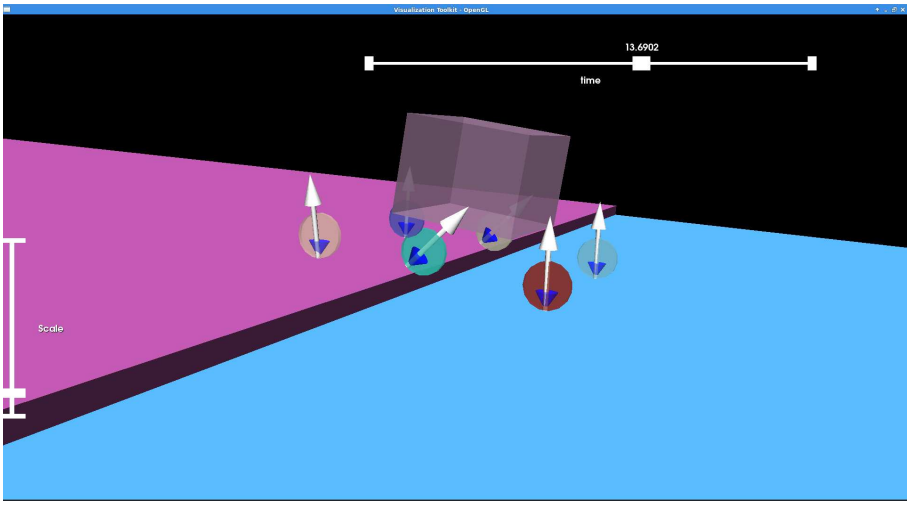


Figure 30: Seventh test scenario (white arrows depict the reaction contact forces)

In this case, following essential quantities have been plotted:

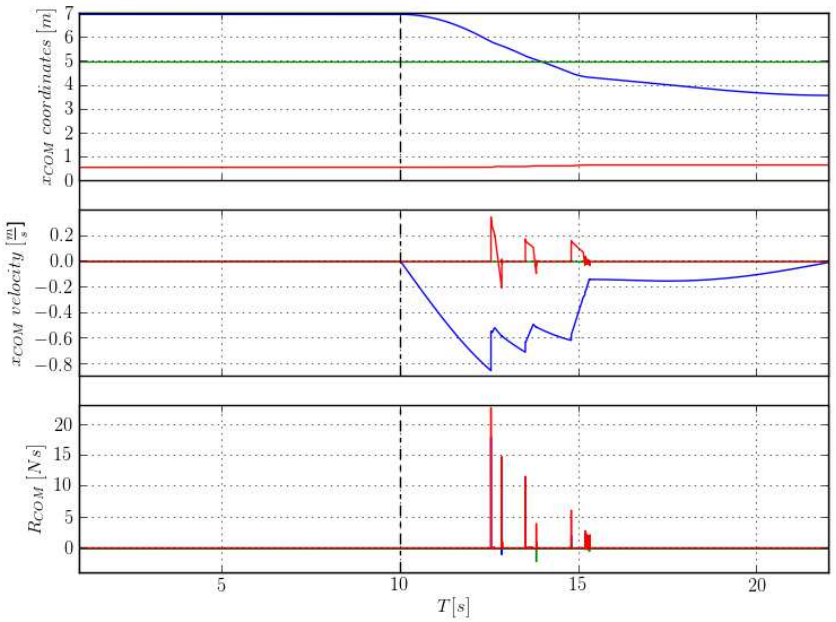
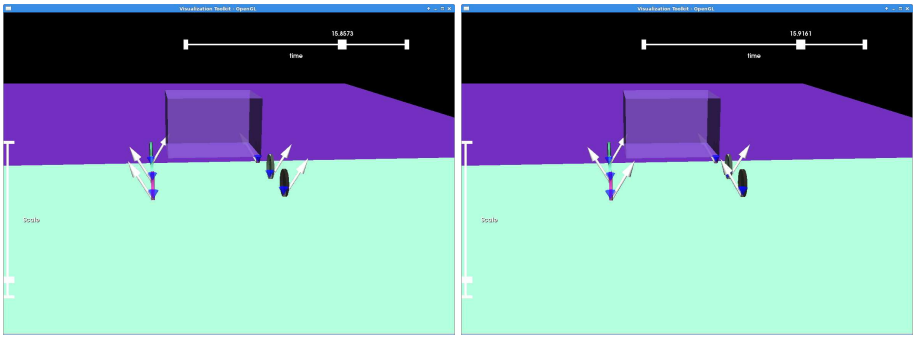


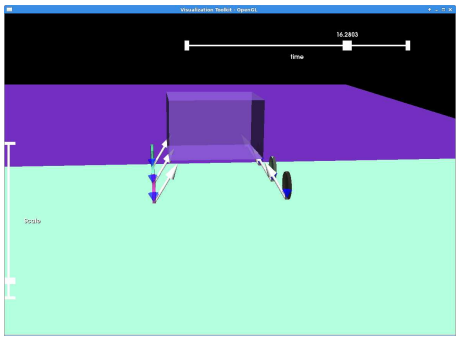
Figure 31: x_{COM} - mass center coordinates

Comments

In the last setting the rover mounts a vertical step. Crossing of the obstacle occurs around 13^{th} second of the simulation then until the end of the simulation rover keeps moving on the upper plane. In the figure 31 one can see detailed motion of the rover when ascending the step along all three coordinates of the mass center. In the same figure, in the two last subplots one notices series of impacts corresponding to the impacts. Namely, in the second sub-plot three impacts can be seen along the z coordinate which correspond to impacts of each pair of wheels. Impacts are also seen in the third sub-plot depicting reaction forces acting on the mass center. Interesting remark can be made about different configurations of contact forces seen in the figure 32. Namely, due to the hyperstatic nature of the simulated system a nonuniqueness of solutions to the friction-contact problem exists. Hence, one can observe different configurations of arrows representing contact forces in the figure 32. This phenomenon can also be seen in figures 20 and 24 in tests 4 and 5 where the solutions oscillate between different values.



(a) Configuration 1 (b) Configuration 2



(c) Configuration 3

Figure 32: Rover climbing the step

Contents

References

[1] V. Acary. Projected event-capturing time-stepping schemes for nonsmooth mechanical systems with unilateral contact and coulomb's friction. *Computer Methods in Applied Mechanics and Engineering*, 256:224 – 250, 2013.

[2] V. Acary, M. Brémond, J. Michalczyk, and R. Pissard-Gibollet. Mechanical simulation of the exomars rover using siconos in 3drov. In *ASTRA 2013 - 12th Symposium on Advanced Space Technologies in Robotics and Automation*, Noordwijk, Netherlands, May 2013.

[3] V. Acary and B. Brogliato. *Numerical methods for nonsmooth dynamical systems: applications in mechanics and electronics*, volume 35. Springer, 2008.

[4] V. Acary and F. Cadoux. *Recent Advances in Contact Mechanics, Stavroulakis, Georgios E. (Ed.)*, volume 56 of *Lecture Notes in Applied and Computational Mechanics*, chapter Applications of an existence result for the Coulomb friction problem. Springer Verlag, 2013.

[5] V. Acary, F. Cadoux, C. Lemaréchal, and J. Malick. A formulation of the linear discrete Coulomb friction problem via convex optimization. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 91(2):155–175, 2011.

[6] Q. Z. Chen, V. Acary, G. Virlez, and O. Brüls. A Newmark-Type Integrator for Flexible Systems Considering Nonsmooth Unilateral Constraints. In Peter Eberhard, editor, *The Second Joint International Conference on Multibody System Dynamics - IMSD 2012*, Stuttgart, Germany, March 2012.

[7] Q.-Z. Chen, V. Acary, G. Virlez, and O. Brüls. A nonsmooth generalized-α scheme for flexible multibody systems with unilateral constraints. *International Journal for Numerical Methods in Engineering*, 2012. submitted.

[8] M. Haddouni, V. Acary, and J.D. Beley. Comparison of index-3, index-2 and index-1 dae solvers for nonsmooth multibody systems with unilateral or bilateral constraints. In *Eccomas. MultiBody dynamics 2013.*, July 2013.

[9] E. Hairer, S.P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer, 1993.

[10] supervisor: Vincent Acary Jianhui Yang. Simulation of a mars rover on granular soils. Technical report, INRIA, Erasmus Mundus Master Internship, 2010.

[11] Bullet Physics Library. `http://bulletphysics.org`.

[12] S Michaud, A Gibbesch, T Thueer, A Krebs, C Lee, B Despont, B Schäfer, and R Slade. Development of the exomars chassis and locomotion subsystem. In *9th International Symposium on Artificial Intelligence. Universal City, CA„ USA: Robotics and Automation for Space (i-SAIRAS 2008)*, 2008.

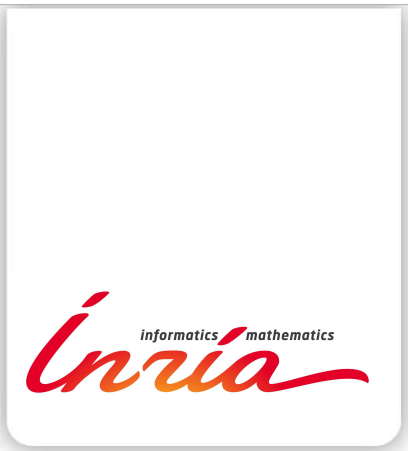
[13] PythonOCC. 3D CAD/CAE/PLM development framework for the Python programming language. `http://www.pythonocc.org`.

[14] F. Radjaï and F. Dubois, editors. *Discrete–element modeling of granular materials*. Iste. John Wiley & Sons, 2011.

[15] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[16] Open CASCADE Technology. `http://www.opencascade.org`.

[17] P.-B. Wieber, F. Billet, R. Boissieux, L.and Pissard-Gibollet, et al. The humans toolbox, a homogenous framework for motion capture, analysis and simulation. In *International Symposium on the 3D Analysis of Human Movement*, 2006.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-0803