

Simulation of a Mars Rover on granular soils



Jianhui Yang

Supervisor: Vincent Acary

Team Bipop

INRIA Rhône Alpes

A internship report submitted for

Erasmus Mundus Master

14 June 2010

Contents

1	Introduction of INRIA Rhône Alpes and Team Bipop	1
1.1	INRIA Rhône Alpes	1
1.2	Team Bipop	2
1.3	Short introduction of the Siconos Platform	2
2	Project description	3
3	Work completed	5
3.1	Starting point: 2D model for a ball with plane	5
3.1.1	Numerical Examples	7
3.2	Extension to 3D model	15
4	Details of model creation and computing	17
4.1	Rover3D model creation with Maple	17
4.1.1	Kinematic data and Geometry description of Rover	17
4.1.2	Dynamical data input	22
5	Contact Description	25
5.1	Computation of distance	25
5.1.1	Wheel/Plane Contact	25
5.1.2	Wheel/Sphere Contact	28
5.2	G function computation	29
5.3	PID controller	30
5.4	Other useful functions	31

6	Numerical Application	32
6.1	Simulation on the Plane	32
6.1.1	Initial condition	32
6.1.2	Simulation result	33
6.2	Simulation on Fixed Spheres	34

List of Figures

3.1	Model	5
3.2	Simulation of a ball on a plan	8
3.3	Reaction force,friction force and velocity of the ball	9
3.4	Simulation of a ball on a Sine curve	9
3.5	Reaction,Friction and velocity	10
3.6	The wheels of the rover on the ground	11
3.7	Local and Global coordinate system for the wheel	12
3.8	Rover on granular soil in 2D	15
4.1	Geometry of Rover	18
4.2	Articular notations of Rover	19
4.3	Part of KinematicData.maple file which describes the position and the orientation of a jointed model with the yaw-pitch-roll angles.	21
4.4	Part of AdditionnalData.maple file which describes the posi- tions of the tags in their attached segment frame.	22
4.5	The DynamicData.maple file describes different inertial param- eters of the model.	23
5.1	Wheel/Plane Contact	26
5.2	Wheel/Sphere Contact	28
6.1	Initial condition of RoverPlane model	32
6.2	Trace of Rover Wheels	33
6.3	Contact Distance between Wheels and Plane	33
6.4	Reaction force of Front Left Wheel	34
6.5	PID controller	34

6.6	Simulation of Rover with fixed Sphere	35
-----	---	----

Chapter 1

Introduction of INRIA Rhône Alpes and Team Bipop

1.1 INRIA Rhône Alpes

INRIA Grenoble - Rhône-Alpes Research Centre is one of the eight research centres run by INRIA, the French National Institute for Research in Computer Science and Control. As a public institute jointly supervised by the French Ministries of Research and of Industry, the centre aims to fulfil two key duties:

- To carry out fundamental and applied research in information and communication science and technology (ICST).
- To share its research results and promote their application in society.

And there are some useful key points for INRIA Rhône Alpes:

- Founded in: 1992
- Approximately 600 people, half of whom are on INRIA's payroll and a third of whom are doctoral students
- 28 research teams
- Research support departments
- Approximately 50% of its budget comes from contracted project funding.

1.2 Team Bipop

Team Bipop is concerned with non-smooth dynamical systems and non-smooth optimization. More precisely, modelling, control and numerical simulation are the main scientific topics. The basic tools therefore come from non-smooth mechanics, systems and control theory, non-smooth optimisation, and convex and non-smooth analysis.

The main applications can be found in mechanical systems (multibody systems with unilateral constraints, friction, nonsmooth contact laws), and in electrical systems (circuits with diodes, MOS transistors). Some more abstract problems (like optimal control with state constraints, generalized predictive control) also fit within this framework.

The main areas of application are: automotive systems, aerospace applications, electro-mechanical systems (mechatronics), robotics, etc. There are still many open fields of theoretical research (in systems theory: controllability, observability, stabilisation, trajectory tracking; in mechanical modelling: multiple impacts modelling, nonmonotone contact laws, Painlevé paradoxes), as well as on a more applied level (numerical simulation and software development). The biped robot is one of the many applications for control and simulation.

1.3 Short introduction of the Siconos Platform

The Siconos Platform is a scientific computing software dedicated to the modeling, simulation, control and analysis of Non Smooth Dynamical Systems (NSDS). Especially, the following classes of NSDS are addressed:

- Mechanical systems with contact, impact and friction
- Electrical circuits with ideal and piecewise linear components
- Differential inclusions and Complementarity systems

Chapter 2

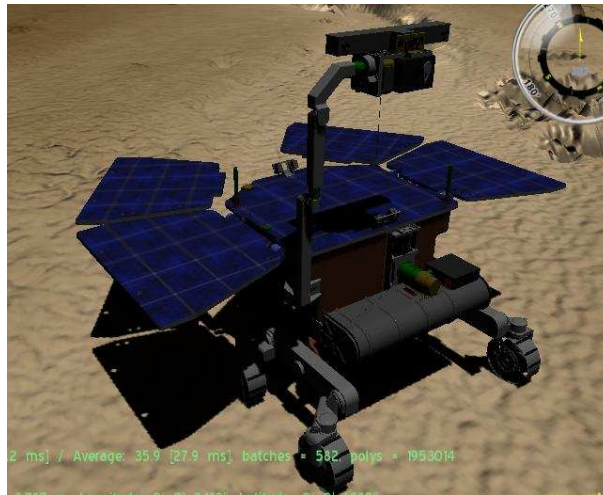
Project description

This internship is taken place within the project Bipop whose research topic is the study of nonsmooth dynamical systems. It is part of a collaboration with the company Trasys contracting the European Space Agency (ESA), in particular, has developed a Trasys 3D simulation of a Mars Rover: 3DROV. Bipop develops a software platform SICONOS for simulation of nonsmooth dynamic systems, especially mechanical systems. Better friction-contact model could be involved in Siconos to improve the existing simulation in 3DROV

The objective of the course internship is to integrate the simulation engine in the tool SICONOS visualization 3DROV ESA and test simulations of moving a Rover Mars on a granular soil.

software

models



objectives

internship

The main work of the intern are:

- Learning to use the platform SICONOS, dynamics and non-regular models of contact friction
- Create ^a 2D model to simulate Rover on granular soil.
- ^{implement a} ~~Make~~ Maple code to generate the dynamical system automatically for ^{the} 3D Rover
- Make simulation for ^{the} 3D Rover model on granular soil.
- Learning the API viewer 3DROV.
- Specification in conjunction with Trasys integration of simulation engine in SICONOS 3DROV coding and integration.
- Experimental simulation of the rover on a granular soil.
- Implement methods of time integration and the sliding contact.

Chapter 3

Work completed

3.1 Starting point: 2D model for a ball with plane

I started with the simplest model, use Siconos to simulate the motion and the contact of a ball falling down on a plane. The Lagrangian dynamical system and the interaction relations are formulated by hand.

Consider the simple case to get used to the code.

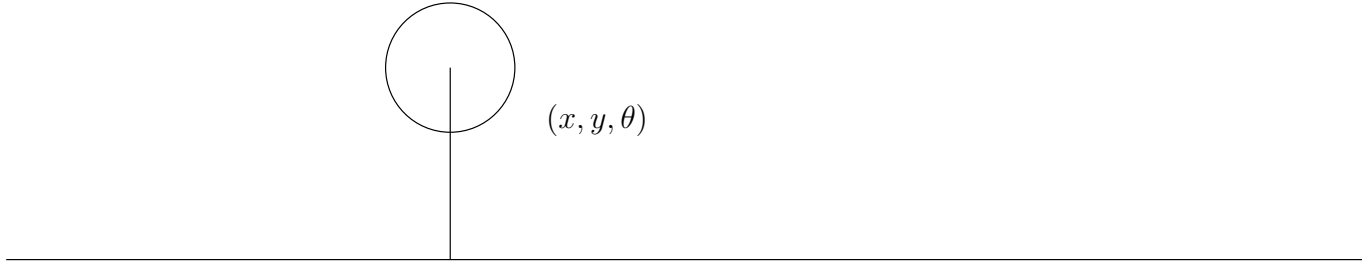


Figure 3.1: Model

Where $q = (x, y, \theta)$ defines the generalized coordinates.

The kinetic energy is

$$K = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}J\dot{\theta}^2 \quad (3.1)$$

3.1 Starting point: 2D model for a ball with plane

In the interaction point between wheel and the ground, we have friction force λ_t , and reaction force λ_n the coordinate of the contact point is $(x+r\sin\theta, y+r\cos\theta)$

In order to transform the force from the local coordinate system to the global coordinate system, the coordinate transformation is used

$$\begin{pmatrix} F_x \\ F_y \end{pmatrix} = \frac{1}{\sqrt{k^2 + 1}} \begin{pmatrix} -k & 1 \\ 1 & k \end{pmatrix} \begin{pmatrix} \lambda_n \\ \lambda_t \end{pmatrix} \quad (3.2)$$

Where, k is the slope of the ground.

We write the generalized forces as:

$$Q_k = F_x \frac{\partial x}{\partial q_k} + F_y \frac{\partial y}{\partial q_k} \quad (3.3)$$

$$\begin{aligned} \begin{pmatrix} R_x \\ R_y \\ R_\theta \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ r\cos\theta & -r\sin\theta \end{pmatrix} \begin{pmatrix} F_x \\ F_y \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ r\cos\theta & -r\sin\theta \end{pmatrix} \frac{1}{\sqrt{k^2 + 1}} \begin{pmatrix} -k & 1 \\ 1 & k \end{pmatrix} \begin{pmatrix} \lambda_n \\ \lambda_t \end{pmatrix} \end{aligned} \quad (3.4)$$

After the calculus of the generalized forces, then we get the Lagrangian equations of motion:

$$\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{x}} \right) - \frac{\partial K}{\partial x} = R_x \quad (3.5)$$

use \displaystyle in equations

$$\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{y}} \right) - \frac{\partial K}{\partial y} = -mg + R_y \quad (3.6)$$

$$\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{\theta}} \right) - \frac{\partial K}{\partial \theta} = R_\theta \quad (3.7)$$

That is,

$$m\ddot{x} = -R_x \quad (3.8)$$

$$m\ddot{y} = -mg + R_y \quad (3.9)$$

$$J\ddot{\theta} = R_\theta \quad (3.10)$$

3.1 Starting point: 2D model for a ball with plane

The formulation of friction contact model is:

$$y = [y_n, y_t]^T, \lambda = [\lambda_n, \lambda_t]^T \quad (3.11)$$

$$\text{if } y_n = 0, \begin{cases} 0 \leq \dot{y}_n \perp \lambda_n \leq 0 \\ \dot{y}_t = 0, \|\lambda_t\| \leq \mu \lambda_n \\ \dot{y}_n \neq 0, \lambda_t = -\mu \lambda_n \text{sign}(\dot{y}_t) \end{cases} \quad (3.12)$$

This model is given in Siconos by the nonsmooth law: `NewtonImpactFrictionNSL`.

In matrix form, we obtain:

$$\begin{pmatrix} m & 0 & 0 \\ 0 & m & \\ 0 & 0 & J \end{pmatrix} \ddot{q} = \begin{pmatrix} 0 \\ -mg \\ 0 \end{pmatrix} + R \quad (3.13)$$

$$R = \begin{pmatrix} R_x \\ R_y \\ R_\theta \end{pmatrix} \quad (3.14)$$

$$q = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \quad (3.15)$$

$$R = H \begin{pmatrix} \lambda_n \\ \lambda_t \end{pmatrix} \quad (3.16)$$

$$H = \frac{1}{\sqrt{k^2 + 1}} \begin{pmatrix} -k & 1 \\ 1 & k \\ -kr \cos \theta - r \sin \theta & r \cos \theta - kr \sin \theta \end{pmatrix} \quad (3.17)$$

$$y_{\text{constraint}}(x, y, \theta) = h(q) = y - kx - b - \frac{r\sqrt{k^2 + 1}}{k} \quad (3.18)$$

3.1.1 Numerical Examples

Here is two simulation results.

($k = -0.2$, and with initial condition $q(t_0) = (3, 10, 0)$ and $\dot{q}(t_0) = (0, 0, 0)$)

3.1 Starting point: 2D model for a ball with plane

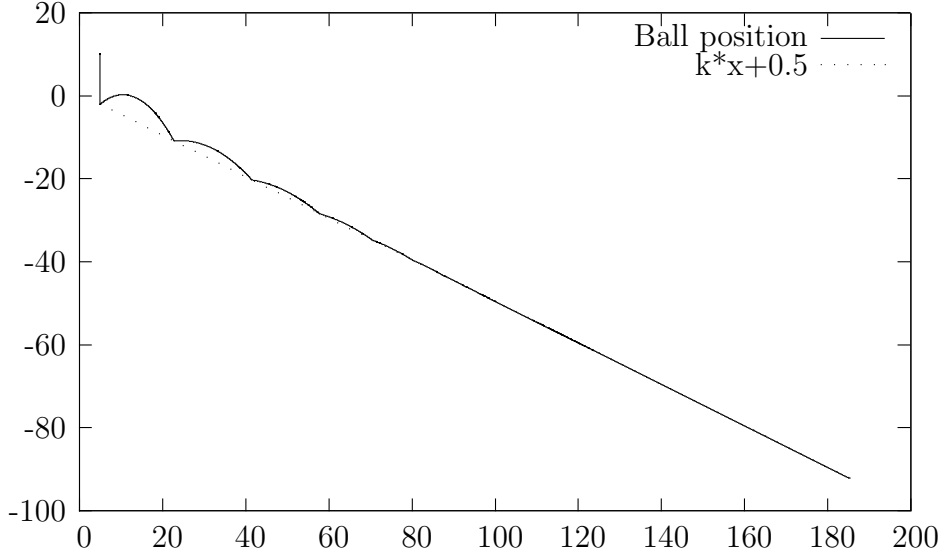


Figure 3.2: Simulation of a ball on a plan

The first example ¹ is the simulation of the motion of a ball falling down over an inclined plane. The mechanical model includes a reaction force and friction between the ball and the plane. The figure (3.2) shows us the movement of the ball along with the time. The figure (3.3) is the simulation result of velocity of the ball, the reaction force, friction force along with the time.

The second case is the ball falling down to a curve surface, with is $y = \sin x$. And still the reaction force and friction are considered in the model.²

Initial condition: $(q(t_0) = (5, 10, 0)$ and $\dot{q}(t_0) = (0, 0, 0)$), coefficient of restitution $e = 0.7$, coefficient of friction $\lambda = 0.5$

¹Source file location: siconos/trunk/SandBox/Rover/2D

²Source file location: siconos/trunk/SandBox/Rover/2D

3.1 Starting point: 2D model for a ball with plane

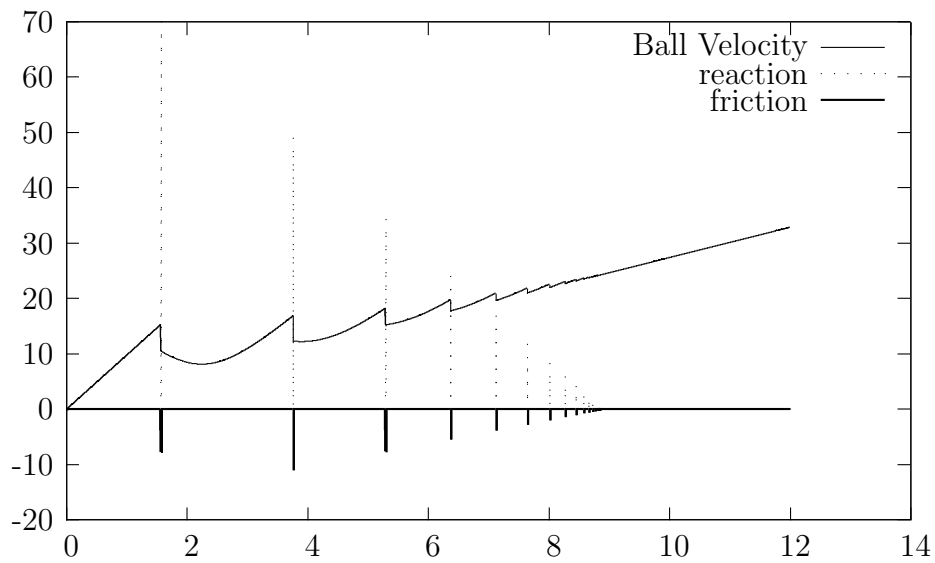


Figure 3.3: Reaction force,friction force and velocity of the ball

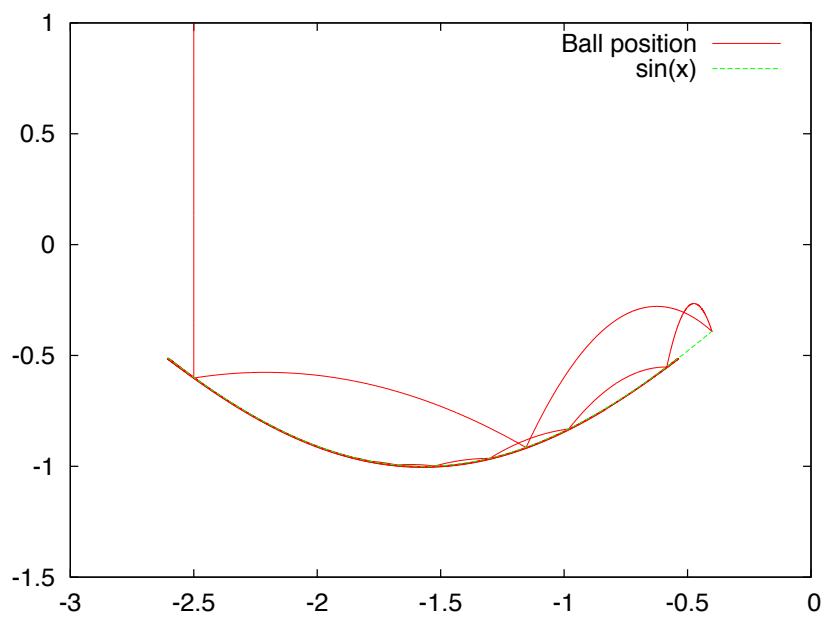


Figure 3.4: Simulation of a ball on a Sine curve

3.1 Starting point: 2D model for a ball with plane

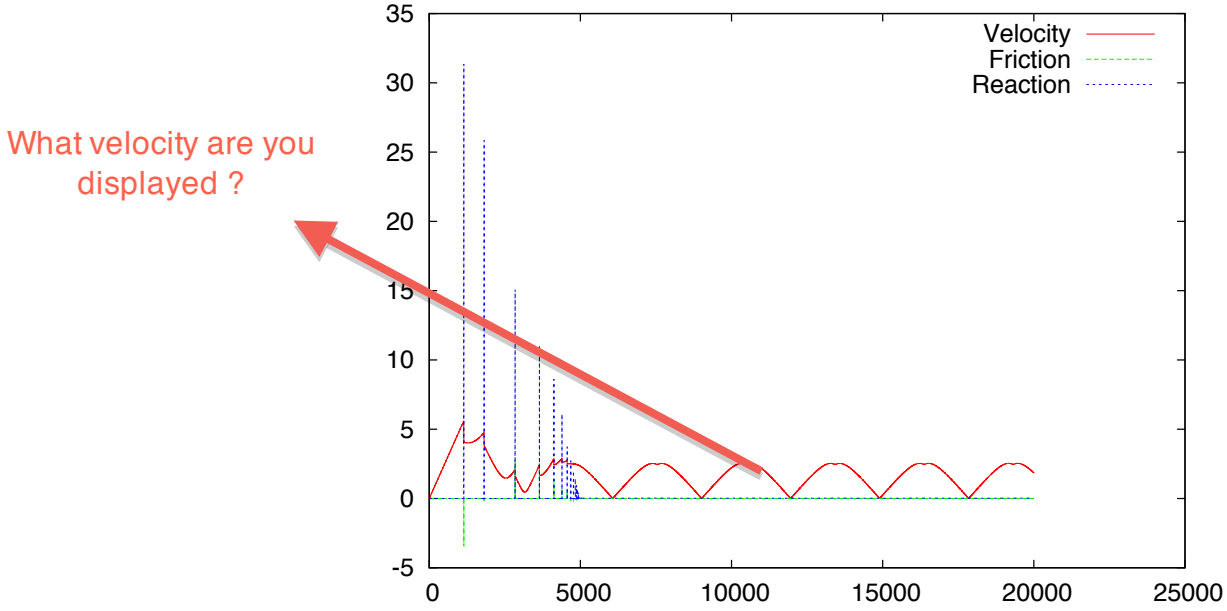


Figure 3.5: Reaction, Friction and velocity

Then we are going to do some more complex cases. We start with a model considering a rover with two wheels, and with friction when a contact happens. We use Lagrange's equations to create the Dynamical system. In this system the number of degree of freedom is 5, we use the generalized coordinates x, y , and $\theta, \theta_A, \theta_B$ to describe the system.

The variables x and y are the location of center of gravity, and θ is the angle of the rod, θ_A, θ_B are the angle of the wheel A and B respectively.

The kinetic Energy of the rod is

$$K_r = \frac{1}{2}m_1(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}J_1\dot{\theta}^2 \quad (3.19)$$

The velocity of the end of the rod is

$$v_{rollerA} = \sqrt{(l\dot{\theta}\cos\theta + \dot{y})^2 + (-l\dot{\theta}\sin\theta + \dot{x})^2} \quad \text{At Point A} \quad (3.20)$$

$$v_{rollerB} = \sqrt{(-l\dot{\theta}\cos\theta + \dot{y})^2 + (l\dot{\theta}\sin\theta + \dot{x})^2} \quad \text{At Point B} \quad (3.21)$$

The kinetic energy and rotation energy of each roller is

3.1 Starting point: 2D model for a ball with plane

$$\frac{1}{2}m_2v_{roller}^2 + \frac{1}{2}J_2\dot{\theta}^2 \quad (3.22)$$

The total kinetic energy of the system is

$$\begin{aligned} K = & \frac{1}{2}m_1(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}J_1\dot{\theta}^2 \\ & + m_2(l^2\dot{\theta}^2 + \dot{y}^2 + \dot{x}^2) + \frac{1}{2}J_2(\dot{\theta}_A^2 + \dot{\theta}_B^2) \end{aligned} \quad (3.23)$$

We name the center of the front wheel "Wheel B", and the center of the back wheel "Wheel A". The gravity center is O. We write the coordinates of these three points with global coordinate parameters $(x, y, \theta, \theta_A, \theta_B)$

$$\begin{aligned} (x_A, y_A) &= (x - l\cos\theta + r\sin\theta_A, y - l\sin\theta + r\cos\theta_A) \\ (x_B, y_B) &= (x + l\cos\theta + r\sin\theta_B, y + l\sin\theta + r\cos\theta_B) \\ (x_O, y_O) &= (x, y) \end{aligned}$$

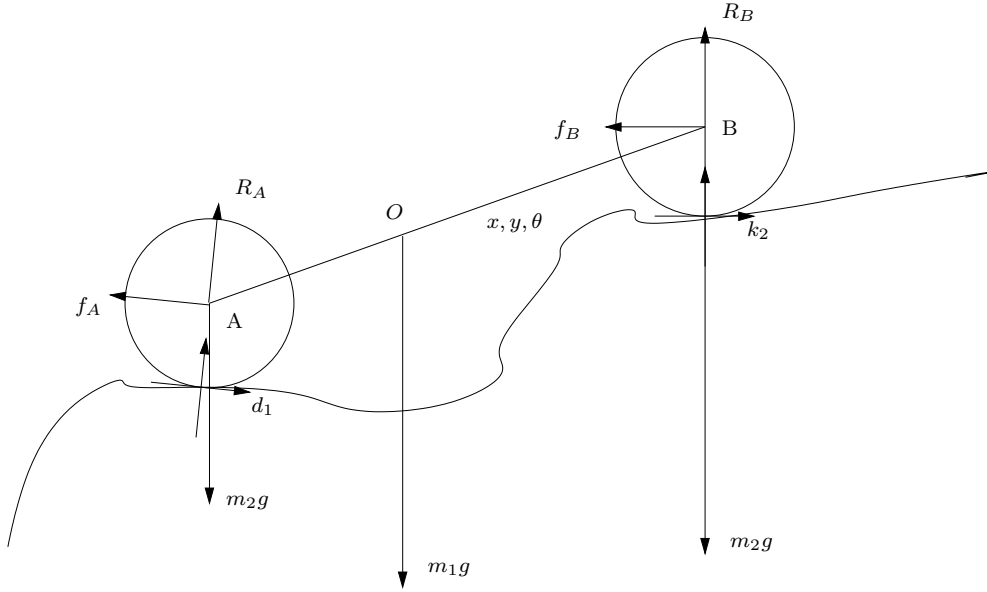


Figure 3.6: The wheels of the rover on the ground

3.1 Starting point: 2D model for a ball with plane

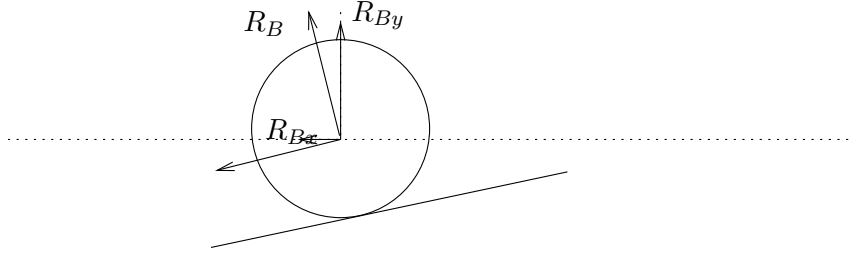


Figure 3.7: Local and Global coordinate system for the wheel

To compute the generalized forces, we use a transformation of the coordinate system and the equation ((3.24)) Let's consider the wheel A for example:

$$Q_k = \sum_{i=1}^n (F_{xi} \frac{\partial x_i}{\partial q_k} + F_{yi} \frac{\partial y_i}{\partial q_k} + F_{zi} \frac{\partial z_i}{\partial q_k}) \quad (3.24)$$

$$\begin{pmatrix} F_{Ax} \\ F_{Ay} \end{pmatrix} = \frac{1}{\sqrt{k_A^2 + 1}} \begin{pmatrix} -k_A & 1 \\ 1 & k_A \end{pmatrix} \begin{pmatrix} \lambda_{An} \\ \lambda_{At} \end{pmatrix} \quad (3.25)$$

$$\begin{pmatrix} R_{Ax} \\ R_{Ay} \\ R_{A\theta} \\ R_{A\theta_A} \\ R_{A\theta_B} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ l \sin \theta & -l \cos \theta \\ r \cos \theta_A & -r \sin \theta_A \\ 0 & 0 \end{pmatrix} \begin{pmatrix} F_{Ax} \\ F_{Ay} \end{pmatrix} \quad (3.26)$$

Let us substitute ~~Equ ((3.25))~~ into ~~Equ ((3.26))~~ we ~~have~~ ^{obtain} the relation from local to global coordinate.

$$\begin{pmatrix} R_{Ax} \\ R_{Ay} \\ R_{A\theta} \\ R_{A\theta_A} \\ R_{A\theta_B} \end{pmatrix} = \frac{1}{\sqrt{k_A^2 + 1}} \begin{pmatrix} -k_A & 1 \\ 1 & k_A \\ -k_A l \sin \theta - l \cos \theta & l \sin \theta - k_A l \cos \theta \\ -k_A r \cos \theta_A - r \sin \theta_A & r \cos \theta_A - k_A r \sin \theta_A \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \lambda_{An} \\ \lambda_{At} \end{pmatrix} \quad (3.27)$$

3.1 Starting point: 2D model for a ball with plane

We use a similar procedure for wheel B

$$\begin{pmatrix} R_{Bx} \\ R_{By} \\ R_{B\theta} \\ R_{B\theta_A} \\ R_{B\theta_B} \end{pmatrix} = \frac{1}{\sqrt{k_B^2 + 1}} \begin{pmatrix} -k_B & 1 \\ 1 & k_B \\ k_B l \sin \theta + l \cos \theta & -l \sin \theta + k_B l \cos \theta \\ 0 & 0 \\ -k_B r \cos \theta_B - r \sin \theta_B & r \cos \theta_B - k_B r \sin \theta_B \end{pmatrix} \begin{pmatrix} \lambda_B^n \\ \lambda_B^t \end{pmatrix} \quad (3.28)$$

The generalized force (without external force) is obtained by:

$$F = \begin{pmatrix} 0 \\ -m_1 - 2m_2 \\ 0 \\ 0 \\ 0 \end{pmatrix} + R_A + R_B \quad (3.29)$$

We ~~can~~ have Lagrange's equations as follows:

$$\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{x}} \right) - \frac{\partial K}{\partial x} = R_{Ax} + R_{Bx} \quad (3.30)$$

use \displaystyle

$$\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{y}} \right) - \frac{\partial K}{\partial y} = -m_1 - 2m_2 + R_{Ay} + R_{By} \quad (3.31)$$

$$\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{\theta}} \right) - \frac{\partial K}{\partial \theta} = R_{A\theta} + R_{B\theta} \quad (3.32)$$

$$\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{\theta}_A} \right) - \frac{\partial K}{\partial \theta_A} = R_{A\theta_A} + R_{B\theta_A} \quad (3.33)$$

$$\frac{d}{dt} \left(\frac{\partial K}{\partial \dot{\theta}_B} \right) - \frac{\partial K}{\partial \theta_B} = R_{A\theta_B} + R_{B\theta_B} \quad (3.34)$$

$$(3.35)$$

We define two functions $d_A(x, y, \theta)$, $d_B(x, y, \theta)$ in the generalized coordinates system to illustrate the nearest distance from the point A and B to the ground surface function $g(x)$

$$\left\{ \begin{array}{l} d_A(x, y, \theta) \geq 0 \\ d_B(x, y, \theta) \geq 0 \\ R_A = 0 \quad \text{if } d_A(x, y, \theta) > 0 \\ R_B = 0 \quad \text{if } d_B(x, y, \theta) > 0 \\ R_A > 0 \quad \text{if } d_A(x, y, \theta) = 0 \\ R_B > 0 \quad \text{if } d_B(x, y, \theta) = 0 \\ f_A = \mu R_A \\ f_B = \mu R_B \end{array} \right. \quad (3.36)$$

3.1 Starting point: 2D model for a ball with plane

Substitute equation(3.23) into equation (3.30),(3.31),(3.33), we have

$$m_1\ddot{x} = R_{Ax} + R_{Bx} \quad (3.37)$$

$$m_1\ddot{y} = -m_1g - 2m_2g + R_{Ay} + R_{By} \quad (3.38)$$

$$J_1 + 2l^2m_2 = R_{A\theta} + R_{B\theta} \quad (3.39)$$

$$J_2\ddot{\theta}_A = R_{A\theta_A} + R_{B\theta_A} \quad (3.40)$$

$$J_2\ddot{\theta}_B = R_{A\theta_B} + R_{B\theta_B} \quad (3.41)$$

In matrix form, we get:

$$q = \begin{pmatrix} x \\ y \\ \theta \\ \theta_A \\ \theta_B \end{pmatrix}, \quad (3.42)$$

$$\begin{pmatrix} m_1 + 2m_2 & 0 & 0 & 0 & 0 \\ 0 & m_1 + 2m_2 & 0 & 0 & 0 \\ 0 & 0 & J_1 + 2l^2m_2 & 0 & 0 \\ 0 & 0 & 0 & J_2 & 0 \\ 0 & 0 & 0 & 0 & J_2 \end{pmatrix} \ddot{q} = F, \quad (3.43)$$

with the relations,

$$h_1(q) = d_A(x, y, \theta) \quad (3.44)$$

$$h_2(q) = d_B(x, y, \theta) \quad (3.45)$$

$$H_1 = \frac{1}{\sqrt{k_A^2 + 1}} \begin{pmatrix} -k_A & 1 \\ 1 & k_A \\ -k_A l \sin \theta - l \cos \theta & l \sin \theta - k_A l \cos \theta \\ -k_A r \cos \theta_A - r \sin \theta_A & r \cos \theta_A - k_A r \sin \theta_A \\ 0 & 0 \end{pmatrix} \quad (3.46)$$

$$H_2 = \frac{1}{\sqrt{k_B^2 + 1}} \begin{pmatrix} -k_B & 1 \\ 1 & k_B \\ k_B l \sin \theta + l \cos \theta & -l \sin \theta + k_B l \cos \theta \\ 0 & 0 \\ -k_B r \cos \theta_B - r \sin \theta_B & r \cos \theta_B - k_B r \sin \theta_B \end{pmatrix} \quad (3.47)$$

With a similar method, we can write the equations for a ball to ball contact, which could describe the contact between the wheel and the sand. Using the

multibody tools in Siconos, we can inject small disks automatically. The multibody tool enable us to simulate the Rover on granular soil in 2D¹ Fig(3.8):

In this case, we use small disk to simulate the granular soil, the relations between the disks will be generated automatically when any two disks are close to each other. ~~And~~ during the simulation, the SpaceFilter class will detect the distance between disks with relations, if they become far from each other, this class will remove the relation, to reduce the memory use and computing ~~cost~~ ^{ational}.

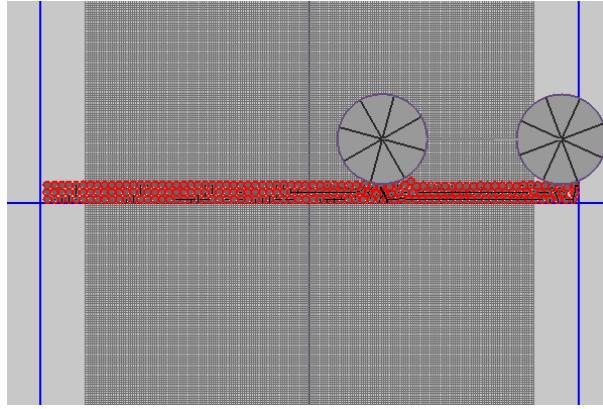


Figure 3.8: Rover on granular soil in 2D

3.2 Extension to 3D model

For 3D Rover modeling, the inertia matrix and contact model interaction relation will be very complex. So, we ~~can~~ write a Maple code to generate this ~~relations~~ ^{ese} automatically, and export into C files. In our Siconos code, we can import this relations as a plug-in function. The Maple code enable us to easily do complex modeling easily with high accuracy.

The software HuMAns developed by team Bipop is used to generate the dynamical model for the Rover. In the software HuMAns, we need to create several maple input files to indicate the relative location of key-points of the model and

¹Source File Location: siconos/trunk/SandBox/Rover/RoverDisks

the degree of freedom of every connecting point.

However, HuMAns is designed for the Robotics dynamic system and contact. In the HuMAns the contact model is for flat plane contact. In the case of Rover, the contact model need to consider more complex contact model. Our simulation should be able to simulate the motion of Rover on uneven ground. So an improvement of HuMAns is required.

A typical dynamic system can be set in the form:

$$M(q)\ddot{q} + N(q, \dot{q})\dot{q} + G(q) = T(q)u$$

where q is the system state variable, $M(q)$ the system inertia, $N(q, \dot{q})$ a non-linear effects matrix, $G(q)$ the forces which derived from a potential energy (for example the gravity), u a command vector and a matrix describing the effect of this command on the system.

When the Rover is moving on the ground, the Rover is support by some contact points between the tyres and the soil. However, so we need to develop a contact model to describe the dynamic of a system with these constraints.

So we have a set of inequalities on the position of the model (the robot or the human), that is, on the state vector q :

$$\varphi_n(q) \geq 0$$

If we consider the points in contact with the environment at time t and if we indicate them with an *, we can write:

$$\varphi_n^*(q) = 0$$

The aim of using HuMAns is to generate the Matrix $M(q)$, $N(q, \dot{q})$, $G(q)$, $T(q)$, $\varphi_n(q)$ as well as $\frac{\partial \varphi_n(q)}{\partial q}$ automatically. Then, this dynamical model could be used in the simulation of Siconos.

Chapter 4

Details of model creation and computing

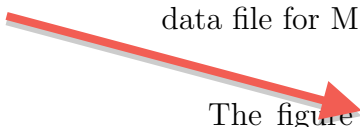
4.1 Rover3D model creation with Maple

A Maple code package will be created to generate the dynamical system of the Rover and C files for visualization.

4.1.1 Kinematic data and Geometry description of Rover

We used the Cardan representation for the Rover model.

The description of the dynamical data of the Rover is composed with "solids". "Solids" are linked with each other using vectors. An (O_k, X_k, Y_k, Z_k) frame is attached to each solid k . The position and the orientation of the frame k is given in the frame ref_k of the solid it is attached to. So we need six parameters to describe a new "solid". A translation (Tx_k, Ty_k, Tz_k) and a rotation of z_k around z_{ref_k} (roll), a rotation of y_k around the resulting y -axis (pitch) and finally a rotation of x_k around the resulting x -axis (yaw) should be included in the Dynamical data file for Maple code.



The figure (??) will show us the geometry of Rover and describe^s what is "solids" and relationship between each other. And part of the **Kinematic-**

Data.maple input file is given to illustrate the structure of the dynamic data description. ¹

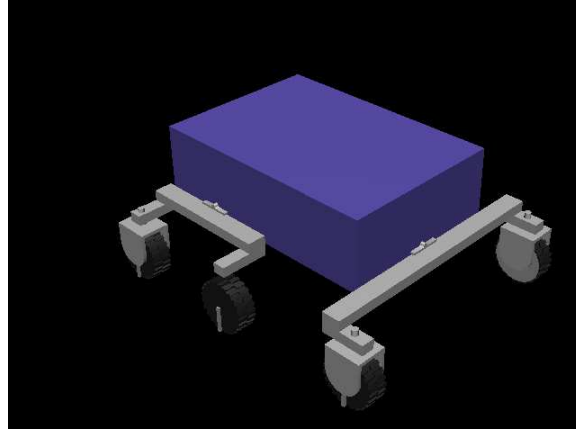


Figure 4.1: Geometry of Rover

With **KinematicData.maple**, the maple code ~~could~~^S know the translation and rotation information between nodes. The table will describe ~~the~~^e all the name of nodes(Solids) and its degree of freedom and location in local frame.

Some characteristic points which we are interested could be defined as tags. We create a maple file **AdditionnalData.maple** to describe the location of Tags. The data describing location tags could be composed as two part:

- The Number of Ref solid the tag is attached to
- One vector describing the translation from this Solid point

The main Maple code **ModelGeneration.maple** will generate a C file **Tags.c** with the data in **AdditionnalData.maple**. This C file enable us to get the gloal coordinate data of all the Tags. In order to use this C file to compute the coordinates. We need to input the state vector q as well as a null matrix T which will be used for output.



¹The 3D model file in Google Sketchup format could found in: /home/bipop/jyang/si-conos/trunk/SandBox/Rover

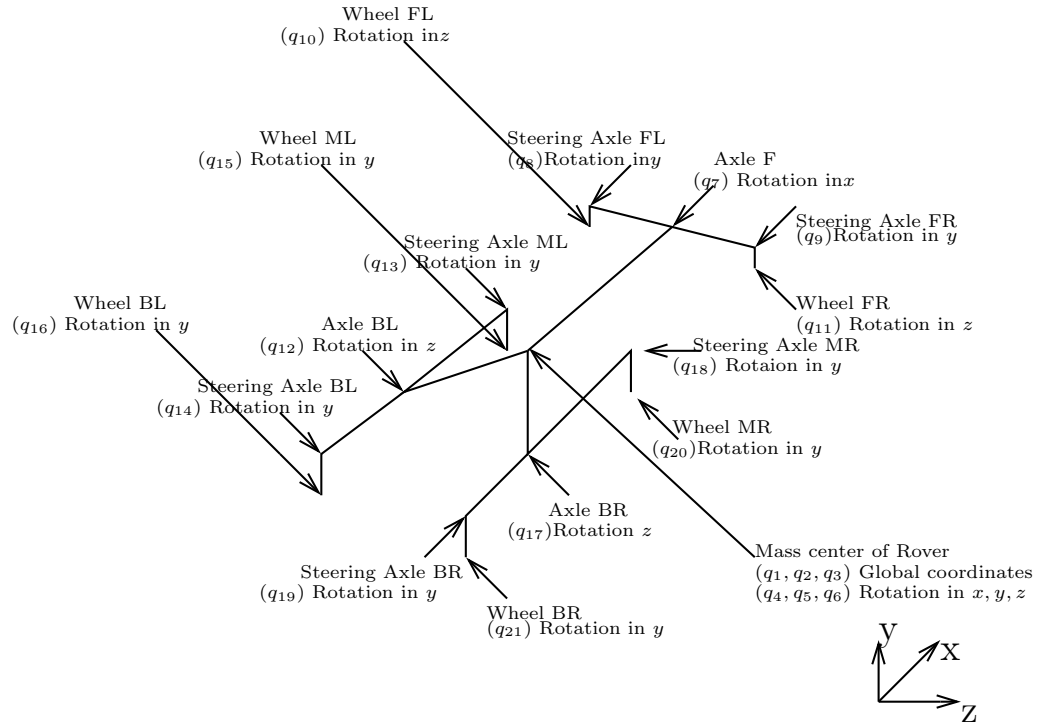


Figure 4.2: Articular notations of Rover

4.1 Rover3D model creation with Maple

No.	Name	Ref. No.	Rotation freedom	Location in Ref fram	Description
1	Mass center	Non	x, y, z	$(0, 0, 0)$	The mass center of the Rover, 6 DOF
2	Axle F	1	x	$(62, -10, 0)$	The pivot at the front
3	Steering Axle FL	2	y	$(0, 0, -53)$	Steering Axle at the front left side
4	Steering Axle FR	2	y	$(0, 0, 53)$	Steering Axle at the front right side
5	Wheel FL	3	z	$(0, -15, 0)$	Rover Tire at the front left side
6	Wheel FR	4	z	$(0, -15, 0)$	Rover Tire at the front right side
7	Axle BL	1	z	$(-40, -10, -53)$	The pivot at left side
8	Steering Axle ML	7	y	$(30, 0, 0)$	Steering Axle at the middle of left side
9	Steering Axle BL	7	y	$(-30, 0, 0)$	Steering Axle at the back of left side
10	Wheel ML	8	z	$(0, -15, 0)$	Rover Tire at the middle of left side
11	Wheel BL	9	z	$(0, -15, 0)$	Rover Tire at the back of left side
12	Axle BR	1	z	$(-40, -10, 53)$	The pivot at right side
13	Steering Axle MR	12	y	$(30, 0, 0)$	Steering Axle at the middle of right side
14	Steering Axle BR	12	y	$(-30, 0, 0)$	Steering Axle at the back of right side
15	Wheel MR	13	z	$(0, -15, 0)$	Rover Tire at the middle of right side
16	Wheel BR	14	z	$(0, -15, 0)$	Rover Tire at the back of right side

Table 4.1: Kinematic Data of the Rover

4.1 Rover3D model creation with Maple

```

# solids number
NSOL := 16:

# degrees of freedom number
NDDL := 21:

#generalized coordinates and velocity definition
q := vector(NDDL):
qdot := vector(NDDL):

# l = vector of mechanical lengths

#Cardan Rotation: We begin with the rotation around z, then around y then
#around x

# Frame 1 : Mass center: Orientation and position
ref_1 := 0:
z_1 := q[6]: #rotation along x,y,z aixe
y_1 := q[5]:
x_1 := q[4]:
Tx_1 := q[1]: #location of the rover
Ty_1 := q[2]:
Tz_1 := q[3]:

# Frame 2 : Axle F (Front)
ref_2 := 1:
z_2 := 0:
y_2 := 0:
x_2 := q[7]: #have a rotation freedom in x direction
Tx_2 := 62: #translation vector from its reference solid (solid 1)
Ty_2 := -10:
Tz_2 := 0:

# Frame 3 : Steering Axle FL (Front Left)
ref_3 := 2:
z_3 := 0:
y_3 := q[8]: #Have a roataion degree of freedom in y direction
x_3 := 0:
Tx_3 := 0: #translation vector from its reference solid(Solid 2)
Ty_3 := 0:
Tz_3 := -53:

.
.
.

# Frame 14 : Steering Axle BR
ref_14 := 12:
z_14 := 0:
y_14 := q[19]:
x_14 := 0:
Tx_14 := -30:
Ty_14 := 0:
Tz_14 := 0:

# Frame 15 : Wheel MR
ref_15 := 13:
z_15 := q[20]:
y_15 := 0:
x_15 := 0:
Tx_15 := 0:
Ty_15 := -15:
Tz_15 := 0:

# Frame 16 : Wheel BR
ref_16 := 14:
z_16 := q[21]:
y_16 := 0:
x_16 := 0:
Tx_16 := 0:
Ty_16 := -15:
Tz_16 := 0:

```

Figure 4.3: Part of **KinematicData.maple** file which describes the position and the orientation of a jointed model with the yaw-pitch-roll angles.

T is on the form:

$$T = \begin{pmatrix} x_{tag1}^0 & y_{tag1}^0 & z_{tag1}^0 \\ x_{tag2}^0 & y_{tag2}^0 & z_{tag2}^0 \\ \vdots & \vdots & \vdots \\ x_{tagNumberOfTags}^0 & y_{tagNumberOfTags}^0 & z_{tagNumberOfTags}^0 \\ x_{CenterOfMass}^0 & y_{CenterOfMass}^0 & z_{CenterOfMass}^0 \end{pmatrix}$$

where the 0 represents the reference frame.

```
# Definition de quelques points importants (tags)

# Tag 1 : Mass center
reftag_1 := 1:           #Ref Number which the tag is attached to
tag_1    := vector([0,0,0]): #vector from this Solid

# Tag 2 : Pivot F
reftag_2 := 2:
tag_2    := vector([0,0,0]):
.
.
.
```

Figure 4.4: Part of **AdditionnalData.maple** file which describes the positions of the tags in their attached segment frame.

4.1.2 Dynamical data input

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) are defined in the file **DynamicData.maple**. The figure 4.5 shows an example of this file.

4.1 Rover3D model creation with Maple

```

#dynamic data for Rover3D

# Gravity vector
Gravity := vector([0, -9.81, 0]):

# solid size matrix (a,b,c) on (x,y,z) in meters unity
sizematrix := [[120,40,100], # trunk: Orientation and position
               [6,6,115],   # Axle F (Front)
               [3,15,0],    # Steering Axle FL (Front Left)
               [3,15,0],    # Steering Axle FR (Front Right)
               [10,0,0],    # Wheel FL
               [10,0,0],    # Wheel FR
               [70,6,6],    # Axle BL
               [3,15,0],    # Steering Axle ML
               [3,15,0],    # Steering Axle BL
               [10,0,0],    # Wheel ML
               [10,0,0],    # Wheel BL
               [70,6,6],    # Axle BR
               [3,15,0],    # Steering Axle MR
               [3,15,0],    # Steering Axle BR
               [10,0,0],    # Wheel MR
               [10,0,0]]:  # Wheel BR

# Rover total mass in kilogrammes unity
Mass := 1:

# solid mass matrix in rate of total mass
massmatrix := [600, # trunk: Orientation and position
               50, # Axle F (Front)
               5, # Steering Axle FL (Front Left)
               5, # Steering Axle FR (Front Right)
               30, # Wheel FL
               30, # Wheel FR
               50, # Axle BL
               5, # Steering Axle ML
               5, # Steering Axle BL
               30, # Wheel ML
               30, # Wheel BL
               50, # Axle BR
               5, # Steering Axle MR
               5, # Steering Axle BR
               30, # Wheel MR
               30]*1: # Wheel BR

MatHuygens := proc(G) option remember;

RETURN(matrix([[G[3]^2+G[2]^2, -G[2]*G[1], -G[3]*G[1]],
               [-G[2]*G[1], G[3]^2+G[1]^2, -G[3]*G[2]],
               [-G[3]*G[1], -G[3]*G[2], G[2]^2+G[1]^2]]));

end:

# function created to compute the inertia matrix

IOMatrix := proc(k) option remember;
local mk, IG, Gk;

mk := Mass * massmatrix[k]:
IG := matrix([[mk*(sizematrix[k,3]^2 + sizematrix[k,2]^2) /12, 0, 0],
              [0, mk*(sizematrix[k,1]^2 + sizematrix[k,3]^2) /12, 0],
              [0, 0, mk*(sizematrix[k,1]^2 + sizematrix[k,2]^2) /12]]):
Gk := G_|| (k):
RETURN(evalm(IG + mk*MatHuygens(Gk))):
end:

.
.
.

#Inertia matrix for Disk
IOMatrixDisk := proc(k) option remember;
local mk, r, IG;

mk := Mass*massmatrix[k]:
r := sizematrix[k,1]:
IG := matrix([[mk*r^2/4, 0, 0],
              [0,mk*r^2/4, 0],
              [0, 0,mk*r^2/2 ]]):
RETURN(IG):
end:

#Inertia matrix for Cylinder
IOMatrixCylinder := proc(k) option remember;
local mk, r, h, IG;

mk :=Mass*massmatrix[k]:
r := sizematrix[k,1]:
h := sizematrix[k,2]:
IG := matrix([[mk*(3*r^2+h^2)/12,0,0],
              [0,mk*r^2/2,0],
              [0,0,mk*(3*r^2+h^2)/12]]):
RETURN(IG):
end:

# Solid 1 : trunk: Orientation and position
m_1 := massmatrix[1]*Mass:
G_1 := vector([0, 0, 0]):
IO_1 := IOMatrix(1):

# Solid 2 : Axle F (Front)
m_2 := massmatrix[2]*Mass:
G_2 := vector([0, 0, 0]):
IO_2 := IOMatrix(2):

# Solid 3 : Steering Axle FL (Front Left)
m_3 := massmatrix[3]*Mass:
G_3 := vector([0, 0, 0]):
IO_3 := IOMatrixCylinder(3):

# Solid 4 : Steering Axle FR (Front Right)
m_4 := massmatrix[4]*Mass:
G_4 := vector([0, 0, 0]):
IO_4 := IOMatrixCylinder(4):

# Solid 5 : Wheel FL
m_5 := massmatrix[5]*Mass:
G_5 := vector([0, 0, 0]):
IO_5 := IOMatrixDisk(5):

# Solid 6 : Wheel FR
m_6 := massmatrix[6]*Mass:
G_6 := vector([0, 0, 0]):
IO_6 := IOMatrixDisk(6):

```

Figure 4.5: The **DynamicData.maple** file describes different inertial parameters of the model.

In this maple file, we could define:

- The gravity vector (defined in vector Gravity)
- The geometry of every segment (defined in Matrix sizematrix, length, height,width,radius etc)
- Mass of each segment (defined in the Matrix massmatrix)
- Mass center of the segment (defined in G_i , i is the solid number), and the function MatHuygens will be called to do the translation if the mass center is not at the geometry center.
- Inertia matrix is defined in the function IOMatrix,IOMatrixDisk,IOMatrixCylinder, which are for inertia matrix computing of rectangle,disk and cylinder respectively.

The **ModelGeneration.maple** file contains:

- **GenerateInertiaMatrix** procedure which uses the **InertiaRecursion** function to generate the **Inertia.c** file.
- **GenerateNLEffectsVector** procedure which uses the **NLEffectsRecursion** function to generate the **NLEffects.c** file

Chapter 5

Contact Description

To describe the contact of the Rover with the solid ground or granular soil in Siconos, we need two plugin functions:

- h function to compute the distance between contact points
- G 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 or plane.

5.1 Computation of distance

5.1.1 Wheel/Plane Contact

With the use of Tags.c file, we can get all the coordinates of tags. In **AdditionalData.maple**, we set some points on the tyres as tags, from which we can get the coordinates. Consider tyre plane contact in Fig (5.1), E is the contact point. Assume the plane equation is $Ax + By + Cz + D = 0$, the length of line EF is the distance we want for the contact model.

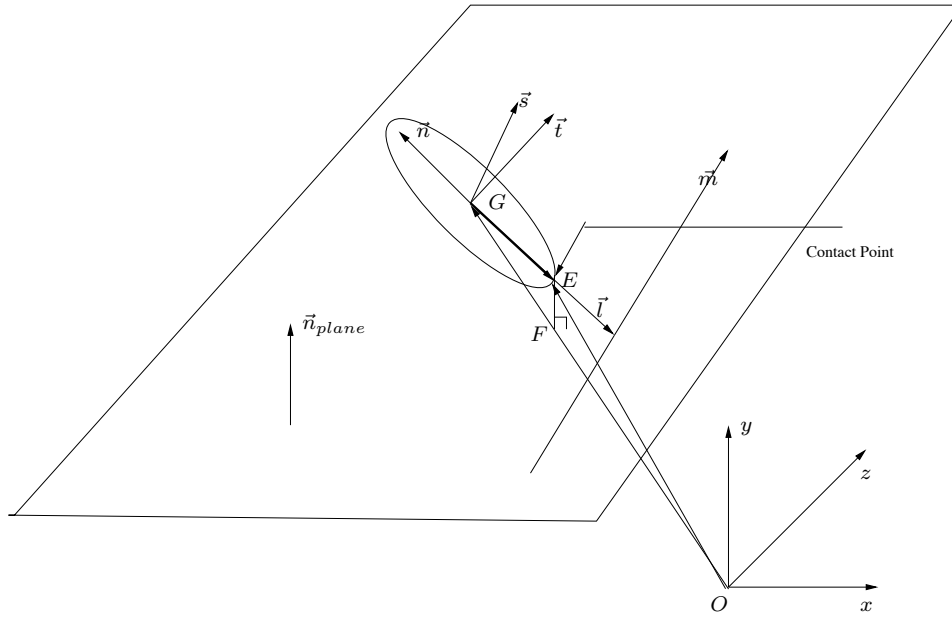


Figure 5.1: Wheel/Plane Contact

The algorithm for computing distance could be summarized as flowing steps.

- Vector \vec{s} is a unit vector perpendicular to the tyre. In order to get this vector, we set this vector as a tag in **AdditionalData.maple**. Then it is possible to obtain this vector from **Tags.c**.
- Vector \vec{n}_{plane} is a vector perpendicular the plane. It is obtained from the plane equation $Ax + By + Cz + D = 0$: $\vec{n}_{plane} = (A, B, C)$
- The vector pointing the moving direction of tyre \vec{m} could be obtained by cross product of \vec{s} and \vec{n}_{plane} , $\vec{m} = \vec{n}_{plane} \times \vec{s}$
- • Vector \vec{t} is the Normalized vector of \vec{m} .
- • The normal vector of loacal fram \vec{n} could be obtained from the cross product of \vec{t} and \vec{s} : $\vec{n} = \vec{s} \times \vec{t}$.
- The vector pointing to the contact point $\overrightarrow{GE} = -R\vec{n}$. Where R is the raduis of the tyre.

- The coordinates of contact point E could be obtain by vector operation:
 $\overrightarrow{OE} = \overrightarrow{OG} + \overrightarrow{GE}$.
- The value of distance could be obtained by using distance formulation (5.1)

$$d = \frac{|Ax + By + Cz + D|}{\sqrt{A^2 + B^2 + C^2}} \quad (5.1)$$

We make a function *ContactDistance* in **ModelGeneration.maple** to compute the distance of six contact points and save the result in a vector. The vector will be export as a C file **Distance.c**. To use this C file, we need to input the state vector q , and four parameter vector $P = (A, B, C, D)$ of the plane, as well as a Null vector *distance* which the result will be saved to. The general format is **Distance(distance,P,q)**.

~~And~~ during the computing of the contact distance, local fram is also obtained: $(\vec{s}, \vec{t}, \vec{n})$.

5.1.2 Wheel/Sphere Contact

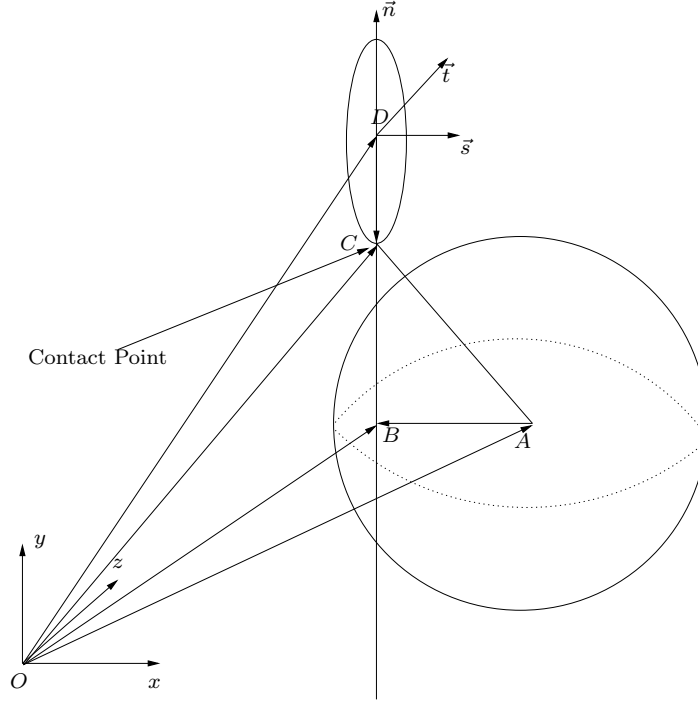


Figure 5.2: Wheel/Sphere Contact

In Wheel/Sphere distance computation, flowing steps are used to obtain the contact distance.

- With similar method used in Wheel Plane contact, we can obtain the vector \vec{s} which is perpendicular to the tyre by defining a tag in **Additional-Data.maple**.
- Assum $\vec{s} = (A, B, C)$ and the coordinates of point D is $\overrightarrow{OD} = (x_0, y_0, z_0)$, we can write the tyre plane equation as $Ax + By + Cz = Ax_0 + By_0 + Cz_0$.
- The distance between the sphere and the tyre plane $\left| \overrightarrow{AB} \right|$ could be obtained by formulation (5.1).

•

$$\overrightarrow{AB} = -\left|\overrightarrow{AB}\right| \vec{s} \quad (5.2)$$

$$\overrightarrow{OB} = \overrightarrow{OA} + \overrightarrow{AB} \quad (5.3)$$

$$\overrightarrow{DB} = \overrightarrow{OB} - \overrightarrow{OD} \quad (5.4)$$

$$\overrightarrow{DC} = \frac{\overrightarrow{DB}}{|\overrightarrow{DB}|} R \quad \text{where } R \text{ is the radius of the tyre} \quad (5.5)$$

$$\overrightarrow{OC} = \overrightarrow{OD} + \overrightarrow{DC} \quad (5.6)$$

- With the coordinates of point C , the contact distance is obtained by $|\overrightarrow{OA} - \overrightarrow{OC}| - R_s$ where R_s is the radius of the sphere.
- What's more, we can obtain the local fram vector $\vec{n} = -\frac{\overrightarrow{DC}}{|\overrightarrow{DC}|}$, $\vec{t} = \vec{n} \times \vec{s}$

5.2 G function computation

To construct the local velocity frame, we need local coordinate system $(\vec{s}, \vec{t}, \vec{n})$

With the local frame $(\vec{s}, \vec{t}, \vec{n})$, we could use matrix $[\vec{s}, \vec{t}, \vec{n}]$ as rotation matrix to transform the global velocity to local frame.

In order to get the velocity in local frame, we start from the global coordinates. Function `ContactVector` in **ModelGeneration.maple** could give the coordinates of contact points for us. They are functions of state vector q

$$P_{global} = \begin{pmatrix} x(q) \\ y(q) \\ z(q) \end{pmatrix} \quad (5.7)$$

And we have velocity in global frame:

$$v_{global} = \frac{dP_{global}}{dt} = \begin{pmatrix} \frac{dx(q)}{dt} \\ \frac{dy(q)}{dt} \\ \frac{dz(q)}{dt} \end{pmatrix} = \begin{pmatrix} \nabla_q x \\ \nabla_q y \\ \nabla_q z \end{pmatrix} \dot{q} \quad (5.8)$$

And we can get the velocity in local frame by multiplying the rotation matrix $T = [\vec{s}, \vec{t}, \vec{n}]$

$$v_{local} = Tv_{global} = T \begin{pmatrix} \nabla_q x \\ \nabla_q y \\ \nabla_q z \end{pmatrix} \dot{q} = G\dot{q} \quad (5.9)$$

The matrix $G = T(\nabla_q x, \nabla_q y, \nabla_q z)^T$ is what we need in Siconos relation definition.

The computing procedure of matrix G is in function `ContactJacobianMatrix` of file **ModelGeneration.maple**

5.3 PID controller

A PID controller(proportionalintegralderivative controller) is used for the steering system of the rover. Siconos enable us to put PID controller easily by plugging a external force term $F_{Ext}(t, z)$ in the Lagrangian Non Linear Dynamical System.

5.10

$$M(q, z)\ddot{q} + NNL(\dot{q}, q, z) + F_{Int}(\dot{q}, q, t, z) = F_{Ext}(t, z) + p \quad (5.10)$$

A typical PID control scheme is composed with three correcting terms in order to minimize the difference between process variable(SP) and setpoint (PV):

$$MV(t) = P_{out} + I_{out} + D_{out} \quad (5.11)$$

Where

$$P_{out} = K_p e(t) \quad (5.12)$$

$$I_{out} = K_i \int_0^t e(\tau) d\tau \quad (5.13)$$

$$D_{out} = K_d \frac{d}{dt} e(t) \quad (5.14)$$

$$K_p, K_i, K_d \quad \text{tuning parameters} \quad (5.15)$$

$$e(t) = SP - PV \quad \text{Error} \quad (5.16)$$

In our particular case, we only use two correcting terms P_{out} and D_{out} :

$$P_{out} = K_p(q(i) - q_0(i)) \quad (5.17)$$

$$D_{out} = K_d(v(i) - v_0(i)) \quad (5.18)$$

Where $q(i)$ is the variable we would like control. $q_0(i)$ is the initial value.

Hence, The dynamical system could be describe^d as:

$$M(q, z)\ddot{q} + NNL(\dot{q}, q, z) + F_{Int}(\dot{q}, q, t, z) = K_p(q(i) - q_0(i)) + K_d(v(i) - v_0(i)) + p \quad (5.19)$$

The code for PID controller is include in the file **RobotPlugin.cpp**

5.4 Other useful functions

To visualize the result in 3D, we need to create a VRML file with the result data. The VRML file use Axis angle rotation representation to describe a rotation, we need to transform our rotation matrix to an axis angle rotation representation when writing the VRML file. This procedure could be found in file **Additional-Data.maple**.

Chapter 6

Numerical Application

6.1 Simulation on the Plane

6.1.1 Initial condition

We consider a Rover on a incline plane. The rover will be put above the plane as the initial position. Then it will fall freely to the plane. The code will simulate the contact and friction between the tyres and the plane. And driving force is put on six wheels. The two front wheel will turn to the right direction.¹

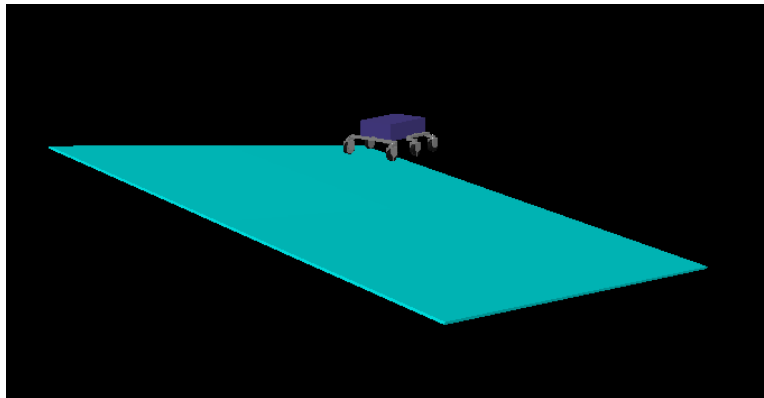


Figure 6.1: Initial condition of RoverPlane model

¹Source File Location: `siconos/trunk/SandBox/Rover/Rover3DPlane`

6.1.2 Simulation result

Beside the VRML animation, we can also export some data to display more details of the movement of the Rover.

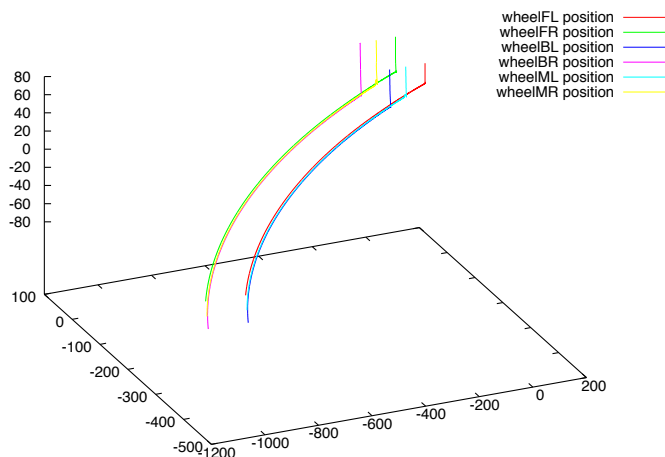


Figure 6.2: Trace of Rover Wheels

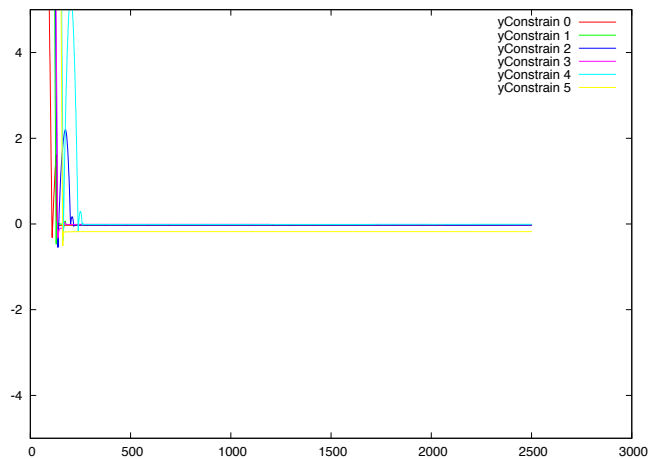


Figure 6.3: Contact Distance between Wheels and Plane

In the figure 6.3, we can find that, all the wheels are constrained upon the

plane very well.

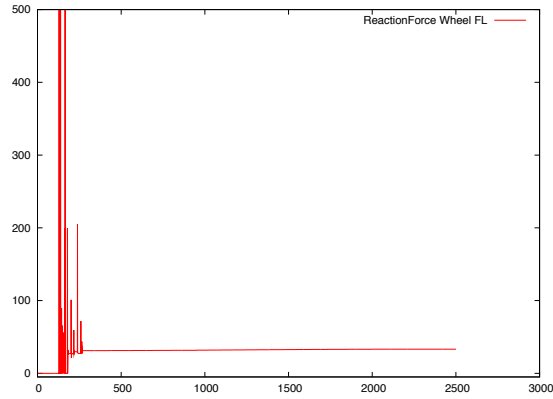


Figure 6.4: Reaction force of Front Left Wheel

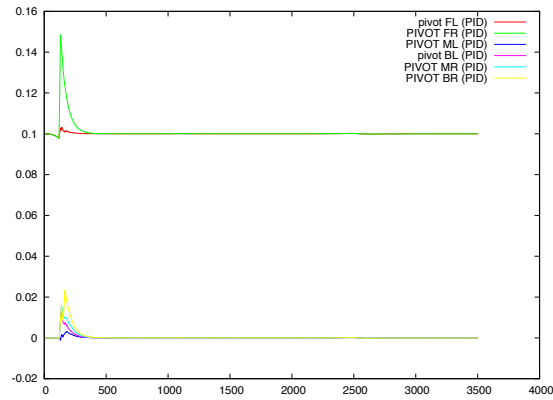


Figure 6.5: PID controller

Fig (6.5) record the angle of steering system along with time. From the curve, we can find that, the PID controller constrain the steering angle very well.

6.2 Simulation on Fixed Spheres

To simulate Mars Rover on the granular soil, we could use spheres to simulate the granular soil. The figure below shows the simulation of Rover with fixed spheres,

to test the validity of our dynamical model.¹

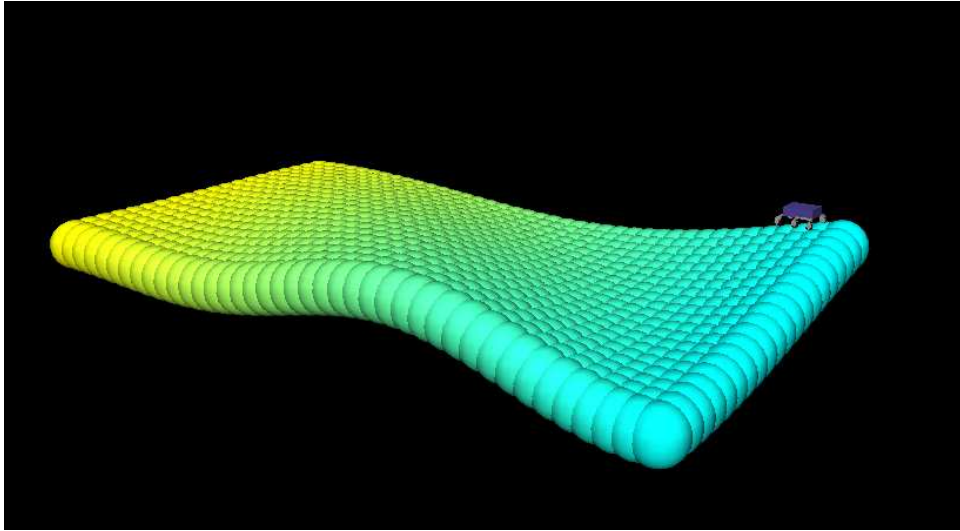


Figure 6.6: Simulation of Rover with fixed Sphere

¹Source Files Location: siconos/trunk/SandBox/Rover/Rover3D_Multi_Sphere