Modelling wheel-soil interactions using the discrete element method for tread shape optimization

Robin Briend

Master of Engineering

Department of Mechanical Engineering

McGill University
Montreal, Québec
August 2010

A Thesis submitted to the Faculty of Graduate studies in partial fulfillment of the degree of Master of Engineering

©Robin Briend, 2010
ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the following individuals and organizations for their constant support and contributions towards my experience at McGill as a Master-thesis student.

I am heartily thankful to my supervisors, Professor Peter Radziszewski and Professor Damiano Pasini, for their encouragement, guidance and support along my research project. It was a real pleasure to benefit from their knowledge in such an emulating atmosphere. I wish to thank the graduate students of their respective groups for their constructive and friendly comments during the weekly group meetings and extra-scholar activities, and all the McGill professors, staff and students I met during the last two years and with whom I had exciting talks and exchanges. A special thank goes to the members of the lunar wheel project (students at McGill and engineers at Neptec and Canadian Space Agency), a team from which I could get advices, motivation and support anytime I needed. The precise review of the present thesis by Suda Martins was also highly appreciated.

I would like to thank my partner in our local EDEM taskforce Nima Gharib for hours of brainstorming and freak outs, and the EDEM support team for answering our innumerable emails providing an eagerly-awaited technical help.

I also have a deep appreciation for the financial and technical assistance provided by NSERC CRD program, Neptec, the Canadian Space Agency, and to DEM Solutions for the EDEM academic license and support.

Finally, I wish to thank all my friends, roommates and relatives in Montreal, francophones and anglophones, with whom I spent two exceptional years in such an enjoyable city.
The structure of a wheel intended for lunar applications requires an innovative design because of the Moon’s specific environment. As in-situ prototype testing is obviously unfeasible, testing can only be conducted on lunar simulant soils, or through simulations. This study presents wheel-soil interaction simulations using the discrete element method (DEM) software EDEM and their use for tread shape optimization.

The DEM parameters of EDEM’s contact-model are first reviewed before presenting a systematic methodology of their calibration. The first step consists in measuring key properties of the real soil with basic experiments and simulating these experiments for different values of the virtual soil’s design variables. The soil’s response surfaces of the targeted properties are then computed, and an optimization algorithm is developed to determine the optimum sets of design variables that minimize the discrepancy of the properties between the real soil and the virtual one.

Then, two different approaches of three-dimensional wheel-soil simulations are described. The first approach involves a displacement-controlled wheel, its tractive performance being measured for various grouser configurations. In the second approach, the wheel is torque-controlled and performances, such as power consumption or speed, are investigated and validated experimentally.

This work proposes a soil simulation and shape optimization tool for the design of a rigid wheel tread that targets a need of the Canadian aerospace industry.
RÉSUMÉ

La structure d’une roue destinée à une application lunaire doit être le fruit d’une conception innovante afin de s’adapter à l’environnement très particulier de la surface de la Lune. Comme les prototypes ne peuvent être testés sur site, des tests sont possibles uniquement sur Terre, sur des simulants de sol lunaire ou en simulations. Dans cette étude, des simulations d’interactions roue-sol utilisant la méthode des éléments distincts (MED) et le logiciel EDEM sont présentées, ainsi que leur utilité dans le cadre de l’optimisation de la géométrie de la bande de roulement.

Tout d’abord, les paramètres MED intervenants dans le modèle de contact de EDEM sont passés en revue et une méthodologie systématique pour leur calibration est proposée. La première étape de cette méthodologie consiste à mesurer des propriétés importantes du sol à modéliser à l’aide d’expériences simples, puis de simuler ces expériences en variant les valeurs des paramètres MED. Les surfaces de réponse pour les propriétés ciblées sont ensuite calculées, et un algorithme d’optimisation détermine les valeurs optimales des paramètres afin de minimiser les différences entre les propriétés du sol réel et celles du sol virtuel.

Ensuite, deux approches de simulations roue-sol tridimensionnelles sont décrites. La première approche implique une roue asservie en déplacements. Les performances en traction de la roue sont mesurées pour différentes configurations de la bande de roulement. Dans la deuxième approche, la roue est contrôlée par un couple moteur, et d’autres performances telles que la puissance consommée ou la vitesse peuvent ainsi être mesurées et validées expérimentalement.

Enfin, les possibilités des simulations proposées en terme d’optimisation de la structure de la bande de roulement de roues rigides sont exposées.
TABLE OF CONTENTS

ACKNOWLEDGEMENTS ......................................................... i
ABSTRACT ........................................................................ ii
RÉSUMÉ ............................................................................. iii
NOMENCLATURE ................................................................. viii
LIST OF TABLES ................................................................. ix
LIST OF FIGURES ............................................................... x
1 Introduction ................................................................. 1
  1.1 Motivation ............................................................... 2
  1.2 Objectives ............................................................... 3
2 Background and literature review ................................. 5
  2.1 Terramechanics .......................................................... 5
    2.1.1 Properties and characterization of a soil ................. 6
    2.1.2 Bekker and Wong theory of ground vehicles applied to the rigid wheel case ......................... 9
    2.1.3 Wheel design concepts for planetary exploration ....... 14
  2.2 The discrete element method ....................................... 16
    2.2.1 Contact models .................................................. 20
    2.2.2 About the software EDEM ................................... 21
  2.3 DEM wheel-soil interaction simulations in the literature ... 22
  2.4 Summary .................................................................. 29
3 Modelling the soil using the discrete element method ....... 32
  3.1 Objectives ............................................................... 32
  3.2 Soil characterization with geotechnical experiments ....... 33
  3.3 Determination of the virtual soil’s design variables ....... 37
    3.3.1 Soil modelling .................................................... 37
    3.3.2 Sensitivity analysis .............................................. 39
3.4 Response surface of the virtual soil properties ........................................ 41
   3.4.1 A review of the Response Surface Methodology (RSM) .......... 42
   3.4.2 Simulation of the direct shear test ........................................ 45
   3.4.3 Simulation of the angle of repose experiment ............................ 48
   3.4.4 Building surrogate models for the virtual soil properties .......... 52
3.5 Search for optimum parameters via the Newton-Gauss algorithm .......... 56
   3.5.1 Basics of multi-objective optimization .................................. 57
   3.5.2 Presentation of the Newton-Gauss optimization algorithm ........... 60
   3.5.3 Results of the calibration problem for lunar regolith and silica
       Barco sand ........................................................................... 61
   3.5.4 Validation of the silica sand calibration with simulations .......... 65
3.6 Summary ....................................................................................... 66

4 Wheel-soil interaction simulations ....................................................... 67
   4.1 Two approaches for wheel-soil simulations using EDEM .............. 67
      4.1.1 Displacement-controlled wheels ........................................ 69
      4.1.2 Torque-controlled wheels ................................................ 73
   4.2 Validation of the results ........................................................... 77
      4.2.1 Experimental validation using a reduced scale vehicle .......... 77
      4.2.2 Comparison to other DEM wheel-soil interaction simulations ... 83
      4.2.3 Comparison to non-DEM based wheel-soil interaction simulations 84
      4.2.4 Comparison to Bekker equations ........................................ 90
   4.3 Tread shape optimization .......................................................... 92
      4.3.1 Existence of optimum grouser configurations ...................... 93
      4.3.2 Building an optimization algorithm for optimum 3D grouser design 97
   4.4 Summary ................................................................................... 103

5 Conclusions and recommendations for future works ......................... 105
   5.1 Conclusions ............................................................................. 105
   5.2 Recommendations for future works ............................................ 107
      5.2.1 Improved soil modelling ................................................... 107
      5.2.2 Enhanced wheel dynamics control ...................................... 110
      5.2.3 Investigating compliant wheel design with FEA-DEM coupling 111

Appendix A: The different contact models in DEM ............................... 115
Appendix B: An example of DEM parameter calibration ......................... 118
Appendix C: About the software EDEM .............................................. 122
Appendix D: Observational error of the angle of repose simulations ....... 127
Appendix E: Results of direct shear test and angle of repose simulations ............ 129
Appendix F: MATLAB function used to compute the angle of repose of a pile of particles .................................................................................................................. 131
Appendix G: Justification of the single sphere particle prototype with angle of repose simulations .............................................................................................. 135
Appendix H: MATLAB function used to build a geometry-wheel made of an assembly of triangles. ............................................................. 138
Appendix I: MATLAB function used to build a particle-wheel made of an assembly of spheres. ............................................................. 150
Appendix J: C++ plug-in used to apply external efforts on the particle wheels in simulations using the torque-controlled approach. ......................... 153
Appendix K: The Newton-Gauss algorithm applied on an analytical example .... 156
References ........................................................................................................... 159
Nomenclature

**Soil properties**

\( \tau_{\text{max}} \)  Maximum shear stress of a soil. [Pa]
\( \varphi \)  Internal angle of friction of a soil.
\( c \)  Cohesion of a soil. [Pa]
\( A \)  Angle of repose of a soil.
\( C_c \)  Compression index of a soil.

**Wheel characteristics**

\( S \)  Contact surface between a structure and a soil. \([m^2]\)
\( W \)  Weight on a wheel. [N]
\( R_R \)  Rolling resistance of the soil on the wheel. [N]
\( H \)  Gross traction effort of the wheel. [N]
\( T \)  Net tractive effort. [N]
\( v \)  Wheel translational velocity. [m/s]
\( \omega \)  Wheel rotational velocity. [1/s]
\( R \)  Wheel radius. [m]
\( w \)  Wheel width. [m]
\( h \)  Grouser height. [m]
\( N_G \)  Number of grousers.
\( \alpha \)  Angle of V-shaped grousers.
\( \beta \)  Angle of the grouser front side with the radial direction.
\( w_b \)  Grouser width at its base. [m]
$w_t$ Grouser width at its top. [m]

**DEM variables**

$\delta t$ Simulation time step. [s]

$T_R$ Rayleigh time step. [s]

$\delta_n$ Normal overlap between contacting particles. [m]

$\delta_t$ Tangential overlap between contacting particles. [m]

$\mathbf{v}_{n}^{\text{rel}}$ Normal relative velocity between contacting particles. [m/s]

$\mathbf{v}_{t}^{\text{rel}}$ Tangential relative velocity between contacting particles. [m/s]

$\nu$ Material property: Poisson’s ratio.

$G$ Material property: shear modulus. [Pa]

$\rho$ Material property: density. [kg/m$^3$]

$\mu_s$ Interaction coefficient of static friction.

$\mu_r$ Interaction coefficient of rolling friction.

$e$ Interaction coefficient of restitution.

$r_0$ Mean particle radius. [m]

$r_0^*$ Mean particle radius constant set to 50\mu m.$
<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2–1</td>
<td>14</td>
</tr>
<tr>
<td>2–2</td>
<td>22</td>
</tr>
<tr>
<td>2–3</td>
<td>31</td>
</tr>
<tr>
<td>3–1</td>
<td>37</td>
</tr>
<tr>
<td>3–2</td>
<td>39</td>
</tr>
<tr>
<td>3–3</td>
<td>41</td>
</tr>
<tr>
<td>3–4</td>
<td>53</td>
</tr>
<tr>
<td>3–5</td>
<td>63</td>
</tr>
<tr>
<td>3–6</td>
<td>64</td>
</tr>
<tr>
<td>E-1</td>
<td>129</td>
</tr>
<tr>
<td>E-2</td>
<td>130</td>
</tr>
<tr>
<td>K-3</td>
<td>158</td>
</tr>
<tr>
<td>K-4</td>
<td>158</td>
</tr>
</tbody>
</table>

**LIST OF TABLES**

Table 2–1 Comparative matrix of wheel design concepts for the Lunar Roving Vehicle [4].

Table 2–2 The material properties and interaction parameters used in EDEM.

Table 2–3 Main characteristics of DEM simulations.

Table 3–1 Regolith and silica sand geotechnical properties.

Table 3–2 Aluminum properties and aluminum-soil interaction parameters.

Table 3–3 Sensitivity analysis of the angle of repose to the seven virtual soil’s design variables.

Table 3–4 Measure of the virtual soil internal angle of friction from direct shear test simulations, with mean particle size $r_0^*$. 

Table 3–5 Predicted virtual soil properties after calibration to fit the silica sand properties.

Table 3–6 Calibration results for different lunar regolith scenarios.

Table E-1 Results of the angle of repose simulations with spherical particles.

Table E-2 Results of the direct shear test simulations with spherical particles.

Table K-3 Example of Newton-Gauss algorithm: Optimum value $x_0$ given as a function of initial guess and tolerance.

Table K-4 Example of Newton-Gauss algorithm: Optimum value $x_0$, with function $\phi_3$ subjected to decreasing weighting factor.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2–1</td>
<td>Pressure distribution in a soil due to a strip load.</td>
<td>7</td>
</tr>
<tr>
<td>2–2</td>
<td>Shear stress as a function of normal stress between two contacting cohesive masses moving along a contact plane.</td>
<td>8</td>
</tr>
<tr>
<td>2–3</td>
<td>The main efforts on a rigid circular wheel.</td>
<td>10</td>
</tr>
<tr>
<td>2–4</td>
<td>The Lunar Roving Vehicle flexible wheel (<em>NASA</em>) and some flexible wheel concepts for ExoMars rovers (<em>ESA</em>).</td>
<td>15</td>
</tr>
<tr>
<td>2–5</td>
<td>The Lunokhod (<em>Wikimedia Commons</em> and Mars Exploration Rovers (<em>NASA</em>)) rigid wheels.</td>
<td>16</td>
</tr>
<tr>
<td>2–6</td>
<td>The DEM loop.</td>
<td>18</td>
</tr>
<tr>
<td>2–7</td>
<td>The linear spring contact model.</td>
<td>20</td>
</tr>
<tr>
<td>2–8</td>
<td>The soil bin with transparent window used by Khot et al. to validate their DEM wheel-soil simulations.</td>
<td>23</td>
</tr>
<tr>
<td>2–9</td>
<td>Experimental and simulated soil deformation under a wheel subject to a 9.8N vertical load.</td>
<td>24</td>
</tr>
<tr>
<td>2–10</td>
<td>Comparison between experimental and simulated wheel dynamic sinkage for different soils and vertical loads.</td>
<td>25</td>
</tr>
<tr>
<td>2–11</td>
<td>Experimental and DEM model groused wheels used by Nakashima.</td>
<td>26</td>
</tr>
<tr>
<td>2–12</td>
<td>Nakashima’s definition of gross traction and motion resistance in DEM wheel-soil simulations.</td>
<td>27</td>
</tr>
<tr>
<td>2–13</td>
<td>Experimental and DEM wheel sinkage, slip and gross traction for a wheel subject to a given load and drawbar load.</td>
<td>28</td>
</tr>
<tr>
<td>2–14</td>
<td>Experimental and DEM wheel net traction as a function of the slip for various grouser height.</td>
<td>29</td>
</tr>
<tr>
<td>2–15</td>
<td>Tractive efficiency of two different wheels as a function of net traction.</td>
<td>30</td>
</tr>
</tbody>
</table>
2–16 Net traction of a wheel as a function of slip, subject to Earth’s and Moon’s 
gravity. ................................................................. 31
2–17 Simulated sinkage of a wheel subject to terrestrial and lunar gravities. .... 31
3–1 Direct shear test experimental setup ................................................. 34
3–2 Shear stress vs. displacement curves from a direct shear test, for various 
pressures. ........................................................................ 35
3–3 Experimental setup of the angle of repose experiment. ......................... 36
3–4 Example of size distribution of the virtual soil particles. ....................... 39
3–5 Sensitivity analysis of the angle of repose to the size distribution parameter δ. 40
3–6 RSM-based optimization workflow. ................................................... 43
3–7 The virtual experimental setup used in the direct shear test simulations. . . 46
3–8 An example of shear force as a function of time extracted from direct shear 
test simulation for various pressure. ........................................... 47
3–9 Linear regression of $\tau_{max}$ as a function of the pressure from direct shear test 
simulations. ........................................................................ 48
3–10 The formation process of a two-dimensional angle of repose simulation. . 49
3–11 The 4 steps of a the 3D angle of repose simulations. ............................ 50
3–12 Angle of repose of the virtual soil as a function of the static and rolling 
friction coefficients $\mu_s$ and $\mu_r$, for a mean particle radius $r_0 = 50\mu m$. .... 51
3–13 Angle of repose of the virtual soil as a function of the mean particle radius 
$r_0$ on a semi-log scale, for static and rolling friction coefficients $\mu_s = 0.5$ 
and $\mu_r = 0.05$. ..................................................................... 56
3–14 Graphic representation of a single-objective optimization problem. ......... 57
3–15 Graphic representation of a multi-objective optimization problem. ........... 59
3–16 Objective function $\phi_4$ as a function of the mean particle radius $r_0$, used in 
the calibration optimization process to express the preference for large 
particles. ........................................................................... 62
4–1 Wheel parametrization used in displacement-controlled CAD wheels. ....... 71
4–2 Close-up on a 16-grouser wheel geometry showing the triangles forming the wheel. ................................................................. 72
4–3 A wheel resting on the soil in the displacement-controlled approach, before the rotation and translation start. ........................................ 72
4–4 An 18-grouser particle wheel made of an assembly of spheres, resting on the soil in a torque-controlled simulation. ........................................... 74
4–5 A simulation using the torque-controlled approach, in which the particle motion is restrained by a frictionless wheel guide. ........................................... 75
4–6 The simple modelling of an electric motor torque map. .......................................... 76
4–7 The Wheely King rover used for experimental power consumption measures, with a close-up on the Mud Thrasher rubber wheels with 18 V-shaped grousers. ..................................................... 78
4–8 Comparison between the real Mud Thrasher wheel and the 4mm-wide particle wheel used in the simulations. ..................................................... 79
4–9 The Wheely King analog simulation system, at two different times. .......... 80
4–10 Energy consumption of a 4mm-wide virtual Mud Thrasher wheel on a silica sand soil. ................................................................. 81
4–11 Comparison between experimental and simulated instant power consumption for a rover with various external load, driven on a flat silica sand test bed at a constant speed of $0.3m/s$. .............................................. 82
4–12 Comparison of 36-grouser wheels subject to the same torque on Earth’s gravity under a 0.3N load, on Moon’s gravity under a 0.3N load, and on Moon’s gravity under a 0.05N load. ..................................................... 84
4–13 Simulated performance comparison of 2 rigid and 8 flexible wheels using a non-DEM based interaction model. .............................................. 85
4–14 Side view of the DEM analog wheel-soil simulation of the ExMR20-3 rigid wheel. ................................................................. 86
4–15 Experimental and simulated average net traction of the ExoMars rover chassis, pulled at a constant speed of $1.08cm/s$, as a function of slip. ... 88
4–16 Simulated net traction of a 32-grouser wheel subject to a 24N load, rolling at $5cm/s$, as a function of slip. .............................................. 89
<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4–17</td>
<td>Example of a wheel experiencing high sinkage and touching the rigid soil box due to high slip.</td>
</tr>
<tr>
<td>4–18</td>
<td>Simulated net traction of a displacement-controlled 32-grouser wheel subject to a 24N load, showing fluctuations due to the presence of grousers.</td>
</tr>
<tr>
<td>4–19</td>
<td>Comparison between the slip estimated with Bekker’s equations and the simulated one, for various loads.</td>
</tr>
<tr>
<td>4–20</td>
<td>Comparison between the sinkage estimated with Bekker’s equations and the simulated one, for various loads.</td>
</tr>
<tr>
<td>4–21</td>
<td>Horizontal position of wheels with various number of grousers, subject to identical polynomial torques, as a function of time.</td>
</tr>
<tr>
<td>4–22</td>
<td>Energy consumption of a 36-grouser wheel to reach the end of the sand box, as a function of the grouser height.</td>
</tr>
<tr>
<td>4–23</td>
<td>Traction of 135mm-diameter displacement-controlled wheels at 5% slip for various grouser configurations.</td>
</tr>
<tr>
<td>4–24</td>
<td>The RSM-based workflow of the automated DEM wheel design optimization loop.</td>
</tr>
<tr>
<td>5–1</td>
<td>The four particle prototypes used by Li et al. to model lunar regolith.</td>
</tr>
<tr>
<td>5–2</td>
<td>The algorithm of the DEM loop including an FEA step for the computation of geometry deformations thanks to an FEA-DEM coupling.</td>
</tr>
<tr>
<td>5–3</td>
<td>An example of DEM-FEA coupling proposed by LEAP Australia and based on the softwares EDEM (DEM part) and ANSYS (FEA part).</td>
</tr>
<tr>
<td>B-4</td>
<td>Corn seed particle model used by Coetzee et al. in 2D DEM simulations.</td>
</tr>
<tr>
<td>B-5</td>
<td>Virtual material’s confined Young’s modulus and internal angle of friction.</td>
</tr>
<tr>
<td>B-6</td>
<td>Comparison of experimental and simulated flow patterns occurring during a silo discharge.</td>
</tr>
<tr>
<td>C-7</td>
<td>An example of multi-spheres particle prototype in EDEM.</td>
</tr>
<tr>
<td>C-8</td>
<td>Definition of material and interaction parameters in EDEM.</td>
</tr>
<tr>
<td>G-9</td>
<td>2D DEM angle of repose simulations of circular and paired (“clump”) particles on Earth and Moon gravities as a function of static friction coefficient, by S. Ji et al.</td>
</tr>
</tbody>
</table>
G-10 3D DEM angle of repose simulations of paired particles with various aspect ratios, as a function of static friction coefficient. .......................... 136
K-11 The function $f(x_1, x_2) = \frac{1}{2} \phi^T \phi$ around its minimum $x_0$. .......................... 157
CHAPTER 1
Introduction

Several space agencies around the world are engaged in planetary exploration programs, aiming at sending probes, rovers, or establishing temporary or permanent bases on the Moon, Mars and even farther celestial bodies. In the Apollo program, the need for a Lunar Roving Vehicle (LRV) was identified just after the first moon landings to increase the range of exploration and capacities of the astronauts on site. Nowadays, mobility systems are even more topical for longer stays involving heavier payloads. As a consequence, the research and development activities in traction systems for extra-terrestrial applications are gaining momentum.

The design of tracked or wheeled traction systems is a multidisciplinary task. It requires a precise knowledge of soil characteristics and mechanical behavior on one hand, and an accurate mastery of mechanical engineering and stress analysis on the other hand. It involves the interaction of both ground and engineering sciences - terramechanics - to identify and meet the required needs and performances of rovers in terms of traction and trafficability on various soils. One major issue in such a design task is that full-scale in situ prototype testing is not possible. Thus, costly analog experiments, involving soil simulants or vacuum chambers, have to be designed to test the successive rover design iterations, in an environment that is as realistic as possible in terms of soil, gravity, atmosphere and temperature conditions.

However, the remarkable progress of computers in the last decades, combined with the advent of new numerical methods, opens new perspectives for soil-structure interaction simulations, giving an alternative to expensive experimental testing and a wider freedom
in the environment setup. The topic of the present study is to define and validate such a numerical wheel-soil interaction modelling methodology, based on the discrete element method.

1.1 Motivation

The Canadian Space Agency (CSA) has commissioned Nepted Design Group the design of a wheeled lunar rover, meeting NASA’s requirements in surface mobility capacities for the future missions on the Moon. Neptec started collaborative projects with academic and industrial partners for the research and development of rover subsystems such as wheels, suspension systems, chassis frame or drive train mechanics. This study is part of one of these collaborative projects, which specific goal is to “define, develop and validate a compliant wheel design methodology which will be used to evaluate and compare the feasibility of different wheel configurations, steering and suspension strategies, and traction designs” [25]. The main parameters to be evaluated are the wheel tractive performance, compliance and modularity. The project team is composed of professors and students at McGill University and engineers from CSA and Neptec. It is stated in the project proposal that six PhD students and two Masters students are supported, focusing on wheel structural design, resistance to wear, prototyping and numerical testing.

The specific topic of the present Master’s thesis concerns numerical testing, and it complements the development of an experimental test bed apparatus used to evaluate different wheel designs. The motivation to develop such a numerical testing methodology is the reduction of the costs and time constraints that are experienced in experimental prototype testing at each iteration of the design process, as well as the possibility to investigate wheel-soil interactions in reduced gravity and on various lunar and martian soils, which is not feasible on a test bed. The numerical method chosen for these interaction
simulations is the discrete element method (DEM), because it is particularly adapted to model granular materials such as the one covering the surface of the Moon.

DEM-based wheel-soil interaction simulations described in the literature are limited to two-dimensional environments and are not seen as design evaluation tools to be part of an optimization loop. In this study, the numerical method is developed to be integrated in a three-dimensional parametrized wheel optimization process, and the overall motivation is to possess, in the medium-term, a DEM analyzer comparable to the FEA or CFD analyzers used in classical design optimization.

1.2 Objectives

As detailed in the project proposal, the present study should address the development and use of the discrete element modeling environment EDEM to simulate different compliant wheels [25]. It is declined in two distinct objectives.

The first objective is to define and develop a numerical modelling and analysis methodology of the wheel design for an optimization purpose. This consists of the following steps: high-fidelity soil modelling, wheel parametrization, wheel-soil interaction simulations and wheel performance evaluation. Only rigid wheels will be considered as EDEM does not support compliant geometries. Thus, the focus will be on the tread shape. A tread parametrization should be proposed to allow design optimization possibilities.

The second objective is to validate the numerical modelling and analysis methodology developed by comparing the simulation results to experimental ones. This is done by modelling known wheel structures, which performances have been previously measured with the test bed apparatus or by other means. The experimental evaluation of the wheels is not part of this study.

This research is presented in the following manner. Chap. 2 reviews the background of terramechanics, soil characterization and discrete element modelling, and describes the
latest publications on DEM wheel-soil interaction simulations. Chap. 3 proposes a systematic methodology for realistic soil modelling, involving the optimization of the soil design variables, based on the response surface methodology. Chap. 4 details the wheel parametrization and motion control, presenting two different simulation approaches based on a displacement-controlled wheel on one hand, and on a torque-controlled wheel on the other hand, and validates the model by comparing simulation results to experimental and theoretical ones. The possibility of using the proposed simulation environment as a wheel design optimization tool is described. Conclusions and recommendations for future works are given in Chap. 5.
CHAPTER 2
Background and literature review

In this chapter, the basics of terramechanics are reviewed, including the soil characterization, the main interactions between the wheel and the soil, followed by the different wheel design concepts used in previous planetary exploration missions. Then, the numerical method used in this thesis to model wheel-soil interaction, called the discrete element method (DEM), is described and its main issues discussed. Finally, two recent papers comparing wheel-soil DEM simulations and experimental results are presented.

2.1 Terramechanics

The first revolution in locomotion and transportation capacities occurred in the ancient times with the invention of the wheel and the harness. Combined with the domestication of the horse, they allowed the towing of heavy payloads on chariots or sleighs. Soon, it became clear that these vehicles were not compatible with muddy or uneven terrain, and the first paved roads appeared. The need for heavier and faster means of transportation resulted in the invention of railways in the sixteenth century, and a “portable” version of it, capable of driving on any type of soil and consisting of a wheel surrounded with mobile wood panels, was patented in 1770 - this was the invention of the tracked vehicle [3]. The technological boom that gave rise to steam and gas engines in the nineteenth century enabled inventors and war offices to develop self-propelled cross-country vehicles at the beginning of the twentieth century. As a result, a new field of study in engineering was born: the characterization of soils and their interactions with locomotion systems such as wheeled or tracked vehicles, namely: terramechanics.
M.G. Bekker is unanimously considered as one of the fathers of terramechanics. He was born in 1905 when the first cross-country vehicles were developed, worked on military tracked vehicles first in Poland, then in Canada during the second world war, and finally switched to lunar vehicle development when hired by General Motors during the Apollo program. In this section of our literature review, we will describe the basics of soil characterization and wheel-soil interactions, mainly based on Bekker’s masterpiece, *Theory of land locomotion* [3].

2.1.1 Properties and characterization of a soil

As the race for heavier and more powerful military off-road vehicles began in the thirties and forties, an investigation on “flotation” and “traction” performances of traction systems on soft grounds revealed the need for a systematic soil characterization. In this terramechanics review, we will follow an inverse approach by first defining soil properties and soil mechanics, and then characterizing vehicle performances and wheel-soil interactions.

The problem of soil mechanics is divided in two parts by Bekker: *elasticity problems* and *stability problems*. The first issue deals with soil deformation under its own load or under static loads such as buildings or dams, considering the soil as an elastic material. No soil failure, or in other words no extensive plastic deformation, occurs in these problems. Stress-strain relations have been investigated in 2D for a semi-infinite ideal soil considered as perfectly elastic, homogeneous and isotropic, and subject to various loads. Although it is far from realistic, this ideal representation of soils obeying to the elastic bodies’ theory gives accurate deformation predictions before breakage through plastic flow. Two interesting results have caught our attention: first, the maximum shear stress is reached at an angle of 45° to the axes of the principal stresses, and second, the result stipulates that the maximum additional pressure in a soil due to a strip load of pressure $P$ and width $w$ is approximately $P/2$ at a depth $w$ and almost negligible at a depth $2w$ (cf. Fig. 2–1). A soil bin of depth
$2w$ or more will then be acceptable in our wheel-soil simulation, as the boundary effect of the rigid bin on the soil will then be negligible.

Figure 2–1: Pressure distribution in a soil due to a strip load of pressure $P$ [3].

On the other hand, *stability problems* define the stress conditions for soil failure by plastic flow. This stability issue is important for our wheel-soil interaction study: indeed, most off-road vehicles experience soil failure when moving on granular soils, leading to plastic deformation of the soil and hence higher wheel sinkage, slip and motion resistance. Soil stability is thus linked to safe vehicle load. Granular masses, which cover most of the trafficable earth surface, have in general *cohesive* and *frictional* properties: the former is responsible for the force bonding two contacting particles free of any external load, while the latter expresses the dependence of the frictional force on the pressure between two particles. The soil properties expressing these aspects are called cohesion $c$ and internal
angle of friction $\varphi$. For two contacting masses subject to a normal load $P$ and moving along a contact plane (cf. Fig. 2–2), Coulomb’s equation gives the shearing stress $\tau$ (shearing force divided by contact area) as a function of $c$ and $\varphi$: $\tau = c + \sigma \tan \varphi$, $\sigma$ being the normal stress ($P$ divided by contact area). The units of $c$, $\tau$ and $\sigma$ are expressed in Pascal.

Figure 2–2: Top: Two cohesive masses A and B moving along a contact plane $m - m$ under a normal load $P$, with a contact surface $S$. Bottom: Shear stress $\tau$ as a function of normal stress $\sigma = P/S$ [3].

The cohesion and the internal angle of friction can be measured using the direct shear test, detailed in Sec. 3.2. Dry sands can be considered as cohesivelss ($c = 0$), while clays or snow can be seen as ideally plastic ($\varphi = 0$). Recommended values for the lunar soil resulting of the Apollo missions are $c = 0.1$ to 1 kPa and $\varphi = 30^\circ$ to 50° [11]. The maximum allowable shear stress in a soil subject to a pressure $p$, before plastic failure, is given by
Mohr-Coulomb equation:

\[ \tau_{\text{max}} = c + p \tan \varphi. \]  

(2.1)

Another interesting characteristic of a soil is its compressibility, or its compaction under compressive stresses. Compressibility is quantified by the compression index \( C_c \), defined as the change of void ratio \( e \) in soil sample occurring when the pressure is increased by an order of magnitude:

\[ C_c = -\frac{\Delta e}{\Delta \log p}. \]  

(2.2)

with the void ratio \( e \) being the ratio of the volume of void with respect to the volume of solid in the soil sample. Compressibility stems from two main mechanisms: particle reorientation at low pressure or for uncompacted soils, and particle deformation or breakage at higher stresses. In terramechanics, it is closely linked to the sinkage of a wheel in the ground. A recompression index \( C_r \), defined as the compression index of a soil previously compacted and then released, can be useful to compute the sinkage of rear wheels rolling in the footprints of front ones. Recommended values for the lunar soil are \( C_c = 0.3 \) and \( C_r = 0.003 \) [11].

Finally, the slope stability of a soil describes how it will recover laterally as a rigid body sinking into it. It is essentially defined by the angle of a slope standing without support, and it depends on the origin of the slope; the slopes of the walls of an excavation will be steeper than a compacted pile of soil, which will be steeper than a simple dumped pile. The angle of the slopes of a dumped pile of soil is called the angle of repose, and a typical value for the lunar regolith is \( A = 50^\circ \) [11].

2.1.2 Bekker and Wong theory of ground vehicles applied to the rigid wheel case

The main interactions between a wheel and a soil are described based on M.G. Bekker and J.Y. Wong [3, 32]. Bekker divides the wheel-soil problem in four distinct cases - the
four possible combinations of a rigid or elastic wheel rolling on a rigid or soft ground. As given in Sec. 2.2.2, the software used here can only simulate interactions between granular materials and rigid geometries. Consequently, the focus is on the rigid wheel on a soft soil case. Moreover, the motion are rectilinear on a horizontal surface, and aerodynamic forces are ignored. In this framework, the equation of motion along the direction of movement $x$ of a wheel subject to a load $W$ is approximated as:

$$m \frac{\partial^2 x}{\partial t^2} = \frac{W}{g} \frac{\partial^2 x}{\partial t^2} = H - R_R - R_d$$

(2.3)

where $H$ is the gross tractive force, $R_R$ the rolling resistance of the soil on the wheel, and $R_d$ an optional drawbar load pulled by the vehicle (cf. Fig. 2–3). The net tractive force $T$, as defined by Bekker, is the difference between gross traction and rolling resistance, or the horizontal component of the total reaction force of the soil on the wheel:

$$T = H - R_R.$$  

(2.4)
The most common approach in wheel design and optimization is to maximize $T$, by maximizing $H$ and minimizing the resistance $R_R$. According to Bekker, “the difficulty in theoretically or experimentally separating $H$ and $R_R$ is the source of most of the misunderstanding in the evaluation of the performance of a driving wheel” [3]. How these two forces are modeled is investigated below.

As a direct consequence of the Mohr-Coulomb equation (Eq. [2.1]), the maximum gross traction of a wheel on a soft soil is characterized by a cohesion $c$, with an internal angle of friction $\varphi$, limited by the shear failure of the soil, occurring for $\tau = \tau_{\text{max}}$:

$$H_{\text{max}} = S\tau_{\text{max}} = Sc + W \tan \varphi$$

(2.5)

where $S$ is the ground contact area. If the torque or force applied on the body is too high, the shear stress in the soil reaches $\tau_{\text{max}}$ and the soil fails, leading to slip. The range of slip values is between 0 (no slip) and 1 (no horizontal motion):

$$s = 1 - \frac{v}{R\omega}.$$ 

(2.6)

In most cases, rigid wheels driving on soft grounds reach shear failure, and thus $H = H_{\text{max}}$, and $s > 0$. For the following, it is assumed that $H = H_{\text{max}}$.

The rolling resistance model is based on the assumption that the pressure $p$ under an object sinking in a soft soil is proportional to the sinkage $z$ to the power $n$, which depends on the soil properties:

$$p = kz^n.$$ 

(2.7)

In most models, $n$ equals to 0, 1/2 (Bernstein’s model) or 1 (Gernster’s, Shultz’s or Grandvoinet’s models) [3]. The work required to compress a unit area to reach a sinkage $z_0$ is:

$$L = \int_0^{z_0} p \, dz = k \frac{z_0^{n+1}}{n+1}.$$ 

(2.8)
Assuming now that the ground compressed by the wheel moves only vertically, the work required by the wheel to compress the soil per unit length traveled is:

\[ R = Lw = kw\frac{z_0^{n+1}}{n+1} \]  

(2.9)

with \( w \) being the wheel width. By assuming the equilibrium of the forces acting on a circular wheel (wheel rolling at a constant speed), subject to a load \( W \) (cf. Fig. 2–3 with \( R_d = 0 \)), Bekker linked the wheel sinkage \( z_0 \) to the load \( W \) with the equation:

\[ W = \frac{kw\sqrt{Dz_0}}{3}z_0^n(3 - n) \]  

(2.10)

with the wheel diameter, \( D \). By eliminating \( z_0 \) from Eqs. [2.9-2.10] and assuming that the rolling resistance \( R_R \) is only due to the force needed to compress the soil, the rolling resistance was expressed as a function of the wheel dimensions, \( W \) and \( k \):

\[
R_R = \begin{cases} 
0.86 \frac{W^{4/3}}{\sqrt{wk}D^{2/3}} & \text{for } n = 1 \\
0.876 \frac{W^{3/2}}{\sqrt{wk^3}D^{3/2}} & \text{for } n = 1/2 \\
\frac{1}{k} \frac{W^2}{wD} & \text{for } n = 0 
\end{cases}
\]  

(2.11-2.12-2.13)

When \( n = 1/2 \), \( k' \) depends on the soil and on \( w \), whereas \( k \) is a function of the soil characteristics only for \( n = 0 \) or 1.

The expression of the net traction in Eq. [2.4] can now be expressed as:

\[ T = (Sc + W \tan \varphi) - f(W, c, \varphi, \Delta) \]  

(2.14)

with \( \Delta \) representing the wheel shape and dimensions and \( f(W, c, \varphi, \Delta) \) the motion resistance. Experimental values of \( k \) have been measured on various soils, and Bekker was then able to compare the net traction of different wheels under a given load numerically.
For example, the difference in net traction of 97cm wheels with 5cm or 10cm grousers was found negligible - as confirmed experimentally. It is noteworthy that the effect of grousers of height $h$ is modelled by Bekker by an increase of the wheel diameter from $D$ to $D' = D + 2h$ in Eqs. [2.13], assuming that the grousers are close enough to each other to clog the soil in between them and thus, augment effectively the wheel diameter. Using a different model, Halkinov assessed the additional thrust given by a grouser of height $h$ vertically sunk in the ground as

$$T_0 = \tau \left( \frac{hw}{\cos \beta \sin \beta} + \frac{h^2}{\cos \beta} \right)$$

(2.15)

with $\tau$ being the shear stress of the soil (cf. Eq. [2.1]), and $\beta = \pi/4 - (\varphi/2)$ [10].

The rolling resistance of multiple wheel carriages was also investigated, and Bekker compared the tractive force of paired wheels in tandem configuration (two wheels on two axles - one front and one rear) or in a dual configuration (two wheels side by side on the same axle). For 4cm-wide wheels, the rolling resistance of the dual configuration was calculated to be 12% higher than the tandem one for a hard sandy road, but 5% lower for a loose sandy soil, for example. However, Bekker cites the experimental work by McKibben and Davidson revealing that in a slightly different configuration, the resistance of the tandem configuration was 23% lower than the dual one on loose sand, and thus invalidating the analytical model [19].

To summarize, although they provide interesting informations about the dependence of $H$ and $R_R$ on the wheel dimensions and load, or about their order of magnitude, these models are not very accurate and are, at times, contradictory. The results of the simulations in Chap. 4 will thus be considered in agreement with the Bekker theory if their order of magnitude and dependence on the main parameters, such as $D$ and $W$, agree with those predicted by the models described in this section.
2.1.3 Wheel design concepts for planetary exploration

In this section, some of the wheel design concepts studied in the last 50 years, for various planetary exploration missions, involving manned or unmanned wheeled rovers, on Mars or the Moon, are presented. As each new mission requires an adapted rover, fulfilling specific criteria, a large variety of design concepts have to be considered. In the selection of the wheel design concept for the Lunar Roving Vehicle (LRV) launched to the Moon in Apollo 15, 16 and 17 missions - the only manned rover vehicle ever used out of Earth -, engineers used a comparative matrix summing the performances of each concept for a wide range of physical or engineering criteria, as shown in Tab. 2–1 [4]. The criteria identified as most important were mechanical reliability, weight, soft ground performance (i.e. traction on flat or sloping terrain) and ride comfort. The wire mesh tire concept

![Table 2–1: Comparative matrix of wheel design concepts for the Lunar Roving Vehicle [4].](image-url)
was found to have the best overall performance, with a total of 706 points, and was thus selected. The LRV wheel builds upon this concept by covering a stiff inner frame with a flexible wire mesh carcass (cf. Fig. 2–4(left)). According to Apollo 15 crew, it fulfilled all expectations: “Traction of the wire wheels was excellent uphill, downhill, and during acceleration (...). Soil accumulation within the wheels was not observed.” [6]. Other deformable wheel concepts involving springs or flexible slats or cylinders in their inner frame were used or are considered for several planetary exploration missions (cf. Fig. 2–4(middle, right)). Although flexible wheels usually develop a higher traction than rigid ones with the same diameter, due to their larger contact surface with the ground, their design and fabrication is much more complex, which make them more fragile. Consequently, rigid wheel concepts are worth investigating.

According to Tab. 2–1, the rigid wheel concept was eliminated from the possible LRV concepts due to its very poor ride comfort. This exclusion can be questioned for unmanned rovers. Indeed, eliminating the ride comfort criterion from the list would rate the rigid wheel second (546.8 points), just below the wire mesh tire (589 points). Another interesting quantity is the soft ground performance of this rigid concept, rated around 50% of the compliant or elastic ones - mainly due to its smaller contact surface with the ground.

![Figure 2–4: (left) The Lunar Roving Vehicle flexible wheel (NASA). (middle, right) Flexible wheel concepts for ExoMars rovers (ESA [24]).](image-url)
leading to a lower maximum traction, as shown in Eq. [2.5]. The main advantages of the rigid wheels lie in their high reliability, stability and environment compatibility. Rigid wheels were used in the Lunokhod rover, an 800kg Russian rover sent to the Moon in the early 1970's (cf. Fig. 2–5)[2], the Mars Exploration Rovers, 185kg rovers launched by NASA in 2003 (cf. Fig. 2–5), and are a possible solution for the ExoMars rover to be sent by ESA in 2018 [28].

In conclusion, the remarks and recommendations of V. Asani about the development of future rover wheels should be kept in mind [2]: the choice of a wheel concept is mission dependent and is a crucial step to the success of building a high performance rover, although the optimization and iterative update of the design concept are the most time-consuming.

2.2 The discrete element method

The dynamic behavior of a particulate system and its interaction with rigid structures or fluid flows is a complex topic, appearing in many engineering problems of various fields, such as mineral processing, pharmaceutical, food processing, or terramechanics. While complex elasticity and structural analysis problems have been solved numerically since the 1950’s with the finite element method (FEM), the modelling of particulate systems has
long remained difficult, mostly because of the huge number of elements they involve and
the complex interactions they bring into play. Indeed, modern discrete element method
(DEM) relies on fairly recent developments. In the middle of the twentieth century, R.D.
Mindlin built a contact model, describing the interaction forces between elastic bodies
in contact with two major articles: Compliance of elastic bodies in contact (1949) and
Elastic spheres in contact under varying oblique forces (1953) [20, 21]. This contact model
is described in detail in Sec. 2.2.1. Thirty years later, P.A. Cundall and O.D.L. Strack
modelled a granular material as an assembly of disks interacting, using Mindlin’s model
(1979) [7]. They simulated the dynamic behavior of their two-dimensional system by
applying Newton’s equations of motion to each one of the disks, computing explicitly
the position, orientation, velocity and forces on each element at each increment of time,
which was basically the inception of DEM. Their validation test, computed on an Interdata
minicomputer, involved only 177 distinct elements, but the immense progress of computers
in the last thirty years allows the modelling of systems with millions of particles, over
millions of time steps presently.

A DEM simulation consists of modelling the evolution over time of an initial system,
formed by a granular material and rigid structures. The initial system is defined entirely in
the pre-processing step: granular material initial positions or creation rules - modelled as an
assembly of distinct particles made of disks (2D simulations) or spheres (3D) of given radii
-, positions of structures and motion rules, gravity, time step (δt), material and interaction
properties. The focus below concentrates on the 3D case. After an initial definition of the
system comes the simulation step.

To simulate the evolution of this system and the interactions between the particulate
material and the structures during one time step, the software runs what could be called
the DEM loop (cf. Fig. 2–6).
This loop in composed of the following steps:

1. From an initial system configuration, consisting of particle and structure positions, orientations and velocities, the first step of the algorithm is the generation of new particles according to particle generation laws defined by the user. This step is optional: it is used to simulate open systems, such as ore conveyors, in which material is fed at a specific location and at a specified flow rate.

2. A list of contacts is built next (particle-particle and particle-structure contacts). Contact between two particles is detected when the distance between two spheres belonging to two different particles gets smaller than the sum of the spheres’ radii.

3. For each contact detected, the contact forces and the resulting moments are computed according to the selected contact model. This step is the core of the DEM loop and
will be explained in more details in the next section. *External forces* are calculated next. Usually, only gravity is considered, but electrostatic forces, drag forces, or customized forces can be added in the case of charged particles, fluid flow, or any external factor specified by the user.

4. Once all forces have been computed, the new position, orientation, velocity and angular velocity of each particle are computed by applying Newton’s equations of motion on them, each one being considered separately. For example, in a 2D case, given a particle position $x(t)$, velocity $\dot{x}(t)$, orientation $\theta(t)$, rotational velocity $\dot{\theta}(t)$, mass $m$ and inertia moment $I$ subject to the resultant force $F$ and resultant moment $M$, they will be updated to:

\[
\begin{align*}
\dot{x}(t + \delta t) &= \dot{x}(t) + \frac{F}{m} \delta t \\
x(t + \delta t) &= x(t) + \dot{x}(t + \delta t) \delta t \\
\dot{\theta}(t + \delta t) &= \dot{\theta}(t) + \frac{M}{I} \delta t \\
\theta(t + \delta t) &= \theta(t) + \dot{\theta}(t + \delta t) \delta t
\end{align*}
\]  

The full details for the 3D case are presented in Cundall and Strack [7]. The structure positions are updated according to the laws of motion specified in the pre-processing step of the simulation.

5. Finally, the system configuration is updated and the *time step is incremented*. The loop can be run continuously, until the simulation duration is reached.

Once the simulation ends, the user can extract the data of interest, such as the number of particles, particles’ positions and velocities, or reaction forces of the material on a surface. Most software packages allow pictures and videos to be exported.
2.2.1 Contact models

The set of equations defining the contact forces between particles, as a function of the particle velocities and overlap, is called a contact model. In this section, the main equations of the first contact model used, defined by Cundall and Strack when they built the discrete element method in 1979 [7], are given. In Appendix A, the model is described with more details and compared to a couple of more sophisticated models.

When they built the DEM in 1979, Cundall and Strack used what is now known as the linear spring contact model. This model connects contacting particles with tangential and normal springs and dampers with respective constants $K_i$ and $C_i$, the indices $i$ referring to tangential or normal directions, as shown in Fig. 2–7. Moreover, the tangential component has a slider in series to illustrate Coulomb’s law of friction, $\mu$ being its friction coefficient.

![Diagram of linear spring contact model](image)

Figure 2–7: The linear spring contact model [23].

Both normal and tangential contact forces are assumed to be composed of an elastic and a damping component, respectively noted: $\vec{F}_n^e$, $\vec{F}_n^d$, $\vec{F}_t^e$ and $\vec{F}_t^d$. In Cundall model, they are given by:

$$\vec{F}_n^e = -K_n \delta_n$$

(2.20)
\[
F_n^d = -C_n v_{rel}^n \\
F_t^e = -K_t \delta_t \\
F_t^l = -C_t v_{rel}^t
\]  
(2.21)  
(2.22)  
(2.23)

where \( \delta_n, \delta_t, v_{rel}^n \) and \( v_{rel}^t \) are respectively the normal and tangential overlaps, and the normal and tangential relative velocities between the two contacting particles. In accordance to Coulomb’s theory, a maximum value for the tangential force is given by:

\[
\left( \overrightarrow{F_t} \right)_{max} = \overrightarrow{F_n} \tan \varphi + c.
\]  
(2.24)

where \( c \) is the cohesion and \( \varphi \) the internal angle of friction.

In summary, the contact forces are computed as a function of material properties such as \( K_n, K_t, C_n, C_t, \varphi \) and \( c \), and the material density \( \rho \) used in Newton’s equations to compute the mass and inertia of the particles (Eqs. [2.16-2.19]).

More complex contact models have been developed since Cundall and Strack’s work (cf. Appendix A), but they all need input parameters reflecting the material properties and behavior. These contact model parameters have to be carefully tuned to ensure that the DEM simulations accurately describe reality. An exemple of DEM input parameter calibration found in the literature is given in Appendix B.

2.2.2 About the software EDEM

The software EDEM 2.2, developed by DEM Solutions (Edinburgh, United Kingdom), was used for all DEM simulations presented in this study. It can run three-dimensional simulations only, as opposed to the DEM works cited in Appendix B and Sec. 2.3. EDEM consists of three main modules:

- **Creator**: the pre-processing module.
- **Simulator**: the computation engine.
• **Analyst**: the post-processing module.

The method to create the simulation environment (in the *Creator* module), the contact model used (in the *Simulator*) as well as the exportation possibilities of the *Analyst* module are fully described in Appendix C. To each geometry or discrete element of a simulation, EDEM assigns a material characterized by 3 properties. Moreover, for each material couple, 3 interaction coefficients are specified. This makes 6 different parameters, listed in Tab. 2–2.

<table>
<thead>
<tr>
<th>Properties associated to material $M_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu$ Poisson’s ratio.</td>
</tr>
<tr>
<td>$G$ Shear modulus. [Pa]</td>
</tr>
<tr>
<td>$\rho$ Density. [kg/m$^3$]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interaction coefficients associated to the material couple $(M_i, M_j)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_s$ Coefficient of static friction.</td>
</tr>
<tr>
<td>$\mu_r$ Coefficient of rolling friction.</td>
</tr>
<tr>
<td>$e$ Coefficient of restitution.</td>
</tr>
</tbody>
</table>

Sec. 2.2.1 highlighted the key role of such parameters in the computation of the contact forces. Thus, a focus is made in this study on EDEM parameter calibration (cf. Sec. 3).

### 2.3 DEM wheel-soil interaction simulations in the literature

In contrast to the Apollo program, contemporary planetary exploration missions planned by the space agencies do not benefit from almost unlimited budget. Extensive prototype testings of numerous wheel design concepts do not fit within the financial constraints. Fortunately, more powerful computers and new methods help engineers designing high-performance rovers. The DEM is one of these new methods, and a few teams try to implement it in their wheel design process. In this section, two of the first papers dealing with DEM wheel-soil interaction simulations are presented. Both of them include 2D wheel-soil simulations and their experimental validation.
In the first paper, written by L.R. Khot et al. in 2007, a circular wheel was driven on a 110cm (length) × 17cm (width) × 10cm (height) soil bin [14]. The motion of the wheel was controlled as follows: the center of the wheel was linked to a rail carriage moving at a constant horizontal velocity, while the rotation of the wheel was controlled by the wheel-soil interactions. An external device applied a vertical load of 4.9N, 9.8N or 14.7N on the center of the wheel. One side of the bin was made of a removable transparent plate, which allowed the printing of a grid on the side of the soil to observe its deformation during the wheel run (cf. Fig. 2–8). Two different soils were used: a coarse sand (soil A) and a finer sand (B), with a mean particle diameter of 4.7mm and 2.0mm respectively. These

![Figure 2–8: The soil bin with transparent window and grid deposition used by Khot et al. to validate their DEM wheel-soil simulations [14].](image)

experiments were modelled in a 2D DEM environment. The virtual soil bin was reduced to 10cm in length and filled with identical circular particles of diameter close to the real mean diameter (990 particles with a 5mm diameter for soil A, 2235 particles with a 3.4mm diameter for soil B). The wheel was circular, and its diameter and load were the same as the experimental ones. The contact model used was the linear spring model.
The validation consisted of two distinct parts: on the one hand, a visual observation of the soil deformation under the wheel was done in both experimental and simulated runs; on the other hand, the wheel sinkage was measured and compared. The observation of soil deformation showed an accumulation of particles in front of the wheel, called “bulldozing effect” (cf. Fig. 2–9).

Figure 2–9: Experimental (left) and simulated (right) soil deformation under a wheel subject to a 9.8N vertical load which brings to light the accumulation of particles in the front of the wheel called bulldozing effect [14].

The wheel sinkage validation was not really satisfactory, as the simulated sinkage was 2 or 3 times higher than the experimental one (cf. Fig. 2–10). According to Khot et al., the reason is that the real soil was compacted before the run whereas the virtual one was not. In the author’s opinion, better results could have been obtained with a higher particle friction coefficient and a different soil modelling, with different particle sizes. Indeed, a soil made of identical circular particles as used in Khot’s simulation cannot be densely compacted. As presented in Sec. 2.1.1, soil compression stems from two mechanisms: particle repositioning and reorientation and particle deformation or breakage. The maximum packing factor of circles in a plane equals $\pi/4 \approx 79\%$ under no load (ideal positioning), and cannot be much higher if compressed by an external force and released, as the particles are supposed to be perfectly elastic (no breakage or deformation).
The origin of the discrepancies between experimental and simulated figures are not clearly identified by Khot et al., who did not question the validity of the DEM parameters. They used friction and damping parameters similar to the ones used in previous DEM studies, without mentioning the calibration method used.

The second interesting paper was written by H. Nakashima et al. in 2007 and aims at optimizing the grouser parameters of a rigid wheel for a future unmanned rover to be launched to the Moon as part of the Japanese SELENE-B mission [23]. The experimental setup consisted in a soil bin filled with the lunar soil simulant MKS-1 develop by the Japanese Space Agency, on which a lugged wheel would roll at a constant rotational velocity $\omega$, with constant vertical load $W$ and drawbar load $R_d$ (cf. Fig. 2–3 for a reminder of the notation). The translational velocity was function of wheel-soil interactions, depending on $\omega$, $W$ and $R_d$. The main wheel parameters were the diameter ($D = 150, 200$ or $250$mm), the number of grousers ($N_G = 18$ or $36$) and the grouser height ($h = 5, 10$ or $20$mm).
In the 2D simulation model, the soil was modelled by 6986 identical circular particles with a diameter of 4mm, initially positioned in an hexagonal lattice, and a wheel modelled as an assembly of circles to mimic the real wheel shape (cf. Fig. 2–11). The linear spring contact model was used, and the DEM parameters were set through a trial-and-error calibration step. The sinkage $z_0$, slip $s$, gross traction $H$, and motion resistance $R_R$ were extracted from the simulations. To identify gross traction and motion resistance separately, Nakashima checked the sign of the horizontal component $f_x$ of the contact forces of particles in contact with the wheel and the wheel itself, as shown in Fig. 2–12. The gross traction was computed as the sum of positive forces (direction of motion), while the motion resistance was the sum of the negative ones:

$$H = \sum f_x^+, \quad R_R = \sum f_x^-$$

The net traction was then computed as $T = H - R_R$. Experimental and simulated results were in good agreement for sinkage, slip and gross traction for given inputs of $W$ and $R_d$ (cf. Fig. 2–13).
Figure 2–12: Definition of gross traction and motion resistance in DEM wheel-soil simulations [23].

By varying the drawbar load $R_d$, while $W$ and $\omega$ remained constant, Nakashima et al. could plot traction vs. slip curves, which are commonly used to characterize the performance of wheels. The results were quite accurate for low slip. The effect of grouser height could be investigated in this framework, revealing possibilities of wheel shape optimization (cf. Fig. 2–14).

Nakashima et al. introduced an interesting tool to characterize wheel’s performance; the traction efficiency $\eta$, defined as the ratio of the drawbar power over the input power, for wheels rolling at constant speeds. The drawbar power is computed as the drawbar load times the velocity, whereas the input power is the product of the angular velocity and the torque (approximated as the net traction times the radius):

$$\eta = \frac{R_d v}{TR\omega} \quad (2.26)$$

This efficiency $\eta$ is plotted for different drawbar loads $R_d$ (i.e. net traction, as $T = R_d$ at constant speed) and different wheel configurations (cf. Fig. 2–15). This comparison is promising for future wheel shape optimization at given mission scenarios.
Figure 2–13: Experimental (left) and DEM (right) wheel sinkage, slip and gross traction for an 18-grouser wheel subject to a 19.6N vertical load and a 4.9N drawbar load [23].

Finally, a virtual comparison of the net traction of a given wheel on Earth \((g = 9.81m/s^2)\) and on the Moon \((g = 1.62m/s^2)\) was conducted, highlighting the worse performances of the wheel on the Moon. Indeed, for a 19.6N load, the net traction was lower and the sinkage was higher under lunar gravity, as shown in Fig. 2–16 and 2–17. It could be argued that the load on the wheel \(W\) should be decreased by the same ratio in the lunar gravity case, as the mass of the rover should remain constant. This might decrease the sinkage difference between terrestrial and lunar cases, and lower even more the lunar net traction, as it is discussed in Chap. 4.
Figure 2–14: Experimental (left) and DEM (right) wheel net traction as a function of the slip for various grouser height. [23].

The virtual particles used by Khot et al. were approximately of the same size as the real soil, whereas Nakashima et al. modelled the lunar simulant MKS-1 with 4-mm particles, which is more than one order of magnitude larger than the real ones. However, Nakashima’s simulation results were closer to the experimental ones. This fact shows that the size of the virtual particles does not have to be realistic to guarantee accurate simulations. The calibration of the DEM parameters, such as friction or damping coefficients, might have a greater impact, and thus should be examined with a greater attention. Only the macroscopic behavior of the soil matters for the simulation of realistic wheel soil interactions.

The main characteristics of the DEM simulations presented by Khot et al. and Nakashima et al. are given in Tab. 2–3 and compared to the characteristics of the simulations presented in this thesis.

2.4 Summary

The twentieth century has seen immense progress in off-road traction systems and vehicles, leading to the birth of a new engineering field: terramechanics. The main wheel-soil interactions, such as traction or rolling resistance, were described semi-empirically as
Figure 2–15: Tractive efficiency of two different wheels as a function of net traction ($L_h$ being the grouser height and $L_n$ the number of grousers) [23].

a function of wheel parameters and soil characteristics, which helped optimize the wheel prototypes used in planetary exploration missions.

As an alternative to costly prototype testing, wheel-soil interaction simulations using the fairly recent discrete element method were described. They were in good agreement with experimental and theoretical results, opening new possibilities for wheel design optimization based on numerical methods.
Figure 2–16: Net traction of a wheel as a function of slip, subject to Earth’s and Moon’s gravity [23].

Table 2–3: Main characteristics of DEM simulations presented in Khot et al. and Nakashima et al. papers and in the present thesis. [23, 14]

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Khot et al.</th>
<th>Nakashima et al.</th>
<th>Present thesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEM environment</td>
<td>2D</td>
<td>2D</td>
<td>3D</td>
</tr>
<tr>
<td>DEM contact model</td>
<td>Linear spring</td>
<td>Linear spring</td>
<td>H-MDns</td>
</tr>
<tr>
<td>Number of particles</td>
<td>up to 2235</td>
<td>up to 7000</td>
<td>up to $\approx$100,000</td>
</tr>
<tr>
<td>Time step</td>
<td>$\approx$15µs</td>
<td>$\approx$100µs</td>
<td>$\approx$1µs</td>
</tr>
<tr>
<td>Wheel motion control</td>
<td>$v$ and $W$</td>
<td>$(\omega, R_d)$ or $(\omega, W)$</td>
<td>Various</td>
</tr>
<tr>
<td>Information extracted</td>
<td>$z_0$</td>
<td>$z_0, T, H, R_R, s$</td>
<td>Depends on motion control</td>
</tr>
</tbody>
</table>

Figure 2–17: Simulated sinkage of a wheel subject to terrestrial (left) and lunar (right) gravities (9.81m/s$^2$ and 1.62m/s$^2$ respectively) [23].
CHAPTER 3
Modelling the soil using the discrete element method

This chapter presents the methodology used to model a granular soil with the discrete element method. The importance of the DEM parameter calibration is highlighted. This guarantees the close match between the real soil macroscopic properties and those of the simulated soil. Two soils are modelled: the lunar regolith and a silica sand used in a single-wheel test bed by Nasim Kaveh-Moghaddam, for experimental validation [13]. The software used in the following is EDEM, developed by DEM Solutions.

3.1 Objectives

As explained in the introduction chapter of this thesis, the goal of this study is to design and optimize a wheel for lunar application. As in-situ prototype testing is impossible, due to the remoteness of the lunar environment, a possible alternative is to use DEM wheel-soil simulations to test different wheel designs and compare their performances. Obviously, these simulations can be useful only if the virtual interactions accurately reflect the real ones. Inaccurate simulated tractive force or sinkage, for instance, might lead to inadequate wheel designs. Consequently, the virtual soil needs to behave like the real one.

As described in Sec. 2.2.2, the behavior of a granular soil modeled by EDEM depends on numerous soil parameters such as static friction coefficient, shear modulus or mean particle radius, which will be designated as soil design variables. The calibration of these parameters is a key step. Without fine tuning, the soil-structure interaction simulations will not quantitatively reflect the real physical phenomena. The set of feasible parameters values is called the soil design space. In this chapter, a systematic method for the calibration
of a granular soil in EDEM, based on the calibration example implemented by Coetzee et al. and outlined in Appendix B, is given. A summary of the different steps of this calibration process is:

I First, the real soil is characterized in Sec. 3.2, by identifying and measuring its geotechnical properties relevant to the study, which is the displacement of a wheel on a deformable soil. Soil properties not related to wheel-soil interactions, such as conductivity or chemical composition for instance, are not considered. The experiments chosen to measure these geotechnical properties are described, and their values for lunar regolith and silica sand are given.

II Next, the virtual soil’s design variables are listed and the sensitivity analysis is carried out to identify the ones having the strongest impact on the soil properties and requiring a fine calibration (cf. Sec. 3.3).

III The geotechnical experiments described in I are then repeated numerically on the virtual soil with various sets of design variables identified in II. From these data, surrogate models are built to describe the response surface of the virtual soil properties as a function of the design variables (cf. Sec. 3.4).

IV Finally, these surrogate models are used to find the optimum design variable set which brings the virtual soil properties as close as possible to the real soil ones. A Newton-Gauss optimization algorithm described in Sec. 3.5 is used.

3.2 Soil characterization with geotechnical experiments

It was previously mentioned that cohesion $c$ and internal angle of friction $\varphi$ are crucial soil properties for the determination of the maximum thrust of a wheel on a granular terrain (cf. Sec. 2.1.1). The Mohr-Coulomb law shows that the maximum shear stress of a soil depends, among other factors, on these two properties (cf. Eq. (2.1)). The direct shear
test allows the simultaneous determination of both \( c \) and \( \varphi \). Its experimental setup uses a three-part box filled with the studied granular material (cf. Fig. 3–1).

![Figure 3–1: The three-part box used in the direct shear test experiment.](image)

A constant load is applied on the top part so that the soil specimen is subject to a constant pressure \( p \). The bottom frame of the box remains stationary, while an increasing longitudinal force \( F \) pushes the upper frame. The longitudinal force \( F \) and the displacement of the upper frame, relative to the bottom one, are recorded. Since the contact between the two frames is frictionless, the longitudinal force \( F \) equals to the product of shear stress in the soil \( \tau \) and the box's cross-section area \( S \): \( F = S\tau \). Starting initially from zero prior to any relative motion of the frames, the longitudinal force \( F \) increases almost linearly as the frames move relatively one along each other. \( F \) ultimately reaches a threshold \( F_{\text{max}} \), which corresponds to the shear failure of the soil on the plane between the two frames: \( \tau_{\text{max}} = F_{\text{max}}/S \). Fig. 3–2 shows an example of shear stress vs. displacement curve.

This experiment is run under various pressures \( p \) by applying different normal loads on the top part of the box, and a linear regression of the maximum shear stresses \( \tau_{\text{max}} \), plotted with respect to \( p \), gives the soil cohesion \( c \) and internal angle of friction \( \varphi \) according to the Mohr-Coulomb equation (cf. Eq. [2.1]).
Since the direct shear test setup involves a small number of parts, it is easy to model. Thus, it is chosen as one of the geotechnical experiments used in the calibration of the virtual soil.

The low cohesion and the high deformability of the lunar soil induce another important phenomenon on the wheel-soil interaction. As a wheel rolling on a granular soil tends to sink, the resistance to motion of the soil on the wheel depends on the way the soil is pushed by the front of the wheel and recovers by the side faces. This avalanching process, occurring in a sloping soil, can be illustrated by the angle of repose experiment. The angle of repose $\alpha$ of a granular material is one of its most explicit properties: it imposes the shape of a heap of gravel or a sand dune for instance (cf. Sec. 2.1.1). However, it is not an intrinsic property of the material and depends on the experimental conditions. Different
experimental setups can be used to measure it, the most common being the slow lifting of a vertical cylinder filled with granular material and initially standing on a plate, or the lifting of one side of a filled box. In this case, the cylinder setup is selected, as used by Ji and Shen in their 2D DEM angle of repose simulations [12]. The angle of repose experimental setup used by Nasim Kaveh-Moghaddam to investigate silica sands is shown in Fig. 3–3 [13].

Figure 3–3: Experimental setup of the angle of repose experiment, consisting of a straw filled with material and lifted upward [13].

In summary, three soil properties were chosen to characterize the soil: cohesion $c$, internal angle of friction $\varphi$ and angle of repose $A$. The values of these properties for a silica Barco sand (from N. Kaveh-Moghaddam direct shear test and angle of repose experiments) and lunar regolith are given in Tab. 3–1 [11].
Table 3–1: Lunar regolith and silica Barco sand properties: cohesion, internal angle of friction and angle of repose.

<table>
<thead>
<tr>
<th></th>
<th>Lunar regolith</th>
<th>Silica Barco sand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cohesion</td>
<td>0.1 to 1 kPa</td>
<td>0.02 kPa</td>
</tr>
<tr>
<td>Internal angle of friction</td>
<td>30° to 50°</td>
<td>26.6°</td>
</tr>
<tr>
<td>Angle of repose</td>
<td>50°</td>
<td>30°</td>
</tr>
</tbody>
</table>

3.3 Determination of the virtual soil’s design variables

This section presents how the virtual soil was modelled in EDEM (Sec. 3.3.1) and how the important design variables describing the virtual soil are identified through a sensitivity analysis (Sec. 3.3.2).

3.3.1 Soil modelling

As described in Sec. 2.2.2, the particles of a granular material are modelled in EDEM from particle prototypes made of spheres. Here, a single sphere particle prototype is used, which means that all particles of the soil will be made from this model, and thus will be spherical. This choice is arguable. The advantage is that, as the simulation run time is proportional to the number of spheres involved, using multi-spheres particles would have lengthened the simulations. For instance, using oblong particles made of three aligned spheres such as the one shown in Fig. C-7 would have resulted in simulation run time multiplied by a factor 3. The drawbacks are that such a soil is less compressible and might not reach high cohesion values, as spheres cannot interlock between each other and can reach an almost-ideally packed configuration very easily (cf. Sec. 2.1.1 for a reminder on the mechanisms involved in soil compression). However, these drawbacks will not be too troublesome, as the soil selected for the experimental validations will be silica sand, which has a low cohesion and low compressibility. For simulations involving high cohesion or highly compressible soils such as lunar regolith, a multi-spheres prototype, or a combination of different prototypes, might be preferred. A justification of the spherical
particle prototype choice is given in Appendix G, where it is shown that the angle of repose of a soil made of oblong particles can be reached with single-sphere particles by tuning the DEM parameters.

The unique particle prototype used is characterized by its material and by its radius $r_0$. As detailed in Tab. 2–2, the material, which will be generically called *soil*, is defined by three design variables (Poisson’s ratio, shear modulus and density), as are the interactions between two soil particles (coefficients of static friction, rolling friction and restitution). This makes a total of seven soil design variables: $r_0$, $\nu$, $G$, $\rho$, $\mu_s$, $\mu_r$ and $e$.

Moreover, the random size option defining the soil particle sizes in the factory settings was used, determining the particle creation rule according to the particle prototype (cf. Sec. 2.2.2). In the random size model, each particle is created as an homothety of the particle prototype with a random ratio $k$, taken between two positive values $k_{\text{min}}$ and $k_{\text{max}}$. In the studied case, the particle prototype is spherical of radius $r_0$, which means that each particle has a radius randomly distributed between $k_{\text{min}}r_0$ and $k_{\text{max}}r_0$. The constraint $k_{\text{max}} + k_{\text{min}} = 2$ is added to ensure that the mean particle radius is $r_0$. A size distribution parameter was defined as: $\delta = k_{\text{max}} / k_{\text{min}}$. Thus, $k_{\text{min}} = 2 / (1 + \delta)$ and $k_{\text{max}} = (2\delta) / (1 + \delta)$. The radius distribution law is then completely determined by the mean radius $r_0$ and the size distribution parameter $\delta$ as:

$$ p(r) = \begin{cases} 
0 & \text{if } r < \frac{2r_0}{1+\delta}, \\
\frac{1}{2r_0 \delta} \frac{\delta+1}{\delta-1} & \text{if } \frac{2r_0}{1+\delta} \leq r \leq \frac{2\delta r_0}{1+\delta}, \\
0 & \text{if } r > \frac{2\delta r_0}{1+\delta}. 
\end{cases} \quad (3.1) $$

which verifies the condition:

$$ \int_0^\infty p(r) \, dr = 1 \quad (3.2) $$
For instance, for \( r_0 = 100\mu m \) and \( \delta = 3 \), the particles radii will be randomly distributed between 50\( \mu m \) and 150\( \mu m \) (cf. Fig. 3–4).

\[
\frac{1}{2r_0} \frac{4+1}{\delta-1} = 10^4 \text{m}^{-1}
\]

Figure 3–4: Uniform distribution of the virtual particles’ radii for \( r_0 = 100\mu m \) and \( \delta = 3 \).

With \( \delta \), the soil is now characterized by eight design variables.

In the simulations, geometries (wheel and soil bin) will be assumed to be made of the same material, \textit{aluminum}. Aluminum properties and soil-aluminum interaction coefficients involved in soil-geometry contacts are taken from EDEM material database and are given in Tab. 3–2. Thus, the addition of geometries in the simulation systems will not bring any extra soil design variable.

Table 3–2: Aluminum properties and aluminum-soil interaction parameters.

<table>
<thead>
<tr>
<th>Aluminum properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson’s ratio.</td>
<td>( \nu = 0.34 )</td>
</tr>
<tr>
<td>Shear modulus.</td>
<td>( G = 25\text{GPa} )</td>
</tr>
<tr>
<td>Density.</td>
<td>( \rho = 2700\text{kg/m}^3 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Aluminum-soil interaction coefficients</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficient of static friction.</td>
<td>( \mu_s = 0.5 )</td>
</tr>
<tr>
<td>Coefficient of rolling friction.</td>
<td>( \mu_r = 0.01 )</td>
</tr>
<tr>
<td>Coefficient of restitution.</td>
<td>( e = 0.5 )</td>
</tr>
</tbody>
</table>

3.3.2 Sensitivity analysis

Although the soil is characterized by eight design variables as mentioned in the previous section, the calibration of each one of them is not feasible since the creation of surrogate models for the virtual soil properties \( c, \varphi \) and \( A \) as a function of these eight variables
would be too time consuming and too complex. Consequently, a screening experiment is conducted to identify which variables have a stronger impact on soil properties and which ones can be neglected. For this sensitivity analysis, the angle of repose simulation, which will be described in details in Sec. 3.4.3, was used.

The impact of the size distribution parameter $\delta$ was investigated first on a given simulation run for $\delta = 1, 1.22, 1.86, 3, and 9$, with the seven other variables kept constant. It was found that $\delta$ had no significant impact on the angle of repose $A$, as shown of Fig. 3–5. Indeed, the variation of $A$ for different values of $\delta$ are smaller than the observational error ($7\%$ for a 90\% confidence interval, as detailed in Sec. 3.4.3). The size distribution parameter was then set to $\delta = 3$ in the following, spreading the particle radii from $0.5r_0$ to $1.5r_0$.

![Figure 3–5: Sensitivity analysis of the angle of repose to the size distribution parameter $\delta$.](image)

The sensitivity analysis to the seven remaining design variables was investigated by first analyzing 5 simulations with the following initial set of parameters:

$$[r_0 = 50\mu m, \ \rho = 3000kg/m^3, \ \nu = 0.2, \ G = 5 \cdot 10^7 Pa, \ \mu_s = 0.3, \ \mu_r = 0.1, \ e = 0.5]$$
The values of the material properties were taken from the literature [11], those of the interaction coefficients from EDEM’s database. The average angle of repose measured was 27.47° (standard deviation: 0.52°). Then the value of one parameter of the initial set was successively changed, while keeping the 6 others constant, and the impact on the angle of repose was measured, as shown in Tab. 3–3.

Table 3–3: Sensitivity analysis of the angle of repose to the seven virtual soil’s design variables.

<table>
<thead>
<tr>
<th>Value of the modified parameter</th>
<th>Angle of repose (average of 3 sim.)</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_0 = 100 \mu m )</td>
<td>26.37°</td>
<td>0.53°</td>
</tr>
<tr>
<td>( \rho = 4000 \text{kg/m}^3 )</td>
<td>27.67°</td>
<td>0.34°</td>
</tr>
<tr>
<td>( \nu = 0.4 )</td>
<td>27.95°</td>
<td>0.28°</td>
</tr>
<tr>
<td>( G = 7 \cdot 10^7 \text{Pa} )</td>
<td>27.06°</td>
<td>0.44°</td>
</tr>
<tr>
<td>( \mu_s = 0.5 )</td>
<td>31.48°</td>
<td>0.64°</td>
</tr>
<tr>
<td>( \mu_r = 0.2 )</td>
<td>36.57°</td>
<td>0.76°</td>
</tr>
<tr>
<td>( e = 0.3 )</td>
<td>27.63°</td>
<td>0.61°</td>
</tr>
<tr>
<td><strong>Initial set:</strong></td>
<td><strong>27.47°</strong></td>
<td><strong>0.52°</strong></td>
</tr>
</tbody>
</table>

The conclusions were that the angle of repose depends mostly on the static and rolling friction coefficients, and to a lesser extent on the mean particle radius. Indeed, the sensitivity derivatives for these three parameters, expressed as the angle of repose variation (\( \Delta A/A \)) over the parameter variation (\( \Delta P/P \), in which \( P \) represents the modified parameter), are 21.9%, 33.1% and −4.0% respectively, whereas there absolute value is below 3% for the four remaining parameters. As a consequence, the three parameters \((r_0, \mu_s, \mu_r)\) are considered as the virtual soil’s design variables to be calibrated, and the response surface for the soil properties are built as their function. The four other variables are kept constant.

### 3.4 Response surface of the virtual soil properties

In order to determine how the DEM parameters describing the virtual soil’s particle prototype influence the macroscopic properties of the soil, the response surface methodology
(RSM) is used. The geotechnical tests presented in Sec. 3.2 are repeated numerically for different design variable sets and surrogate models are built for each one of the three physical properties from the collected data. Before detailing the simulated experiments and the data collection, the principle of RSM and its fields of application are reviewed.

3.4.1 A review of the Response Surface Methodology (RSM)

In The Response Surface Methodology: Process and Product in Optimization Using Designed Experiments [22] published in 1995, R. Mayers and D. Montgomery give a clear and complete overview of the RSM. They describe a “collection of statistical and mathematical technique” used to optimize “some performance measure or quality characteristic of a product or process”. The main purpose is to optimize a measure or characteristic depending on several input variables. In the studied case, it is used to optimize the design of a soil prototype particle, a function of several parameters, or design variables. The optimization is obtained through a minimization (or maximization) of the multi-variable function, or surface, describing the process. The modelling of this function, called surrogate model, is the critical step; it is implemented by repeating a given number of experiments or simulations for different input variable sets in order to map the response of the process. The advantage of such a methodology in an optimization task, compared to a basic trial-and-error approach for example, is to decrease the number of experiments - and save time and money - while enhancing the quality of the optimum set.

RSM is an iterative process, its first step being the identification of the variables characterizing the process to optimize. A sensitivity analysis follows to select the variables with non-negligible effect. This is done by a screening experiment (cf. Sec. 3.3). The variables with higher impact are the only ones considered in the following. The other parameters do not need to be optimized, as they do not significantly affect the result. Finally, boundaries
have to be set for each variable. For the case of a chemical reaction for example, the variables *time* and *mass of reactant* must be positive and below an economically unacceptable limit. This completes the definition of the initial design space, first block on the RSM optimization workflow presented in Fig. 3–6.

Figure 3–6: An RSM-based optimization workflow using surrogate models. The optional loop is possible only for online model management.

Once the design space is defined, a strategy is selected to explore its extent (second block in Fig. 3–6). The surrogate model of the process will be built from the data point collected during this step (third and fourth blocks in Fig. 3–6). The goal here is to find the strategy that would allow the computation of the most accurate surrogate model with a minimum design space exploration. This research strategy is called *Design of Experiment* (DoE). The simplest DoE is the $2^k$ factorial design. It consists in choosing two feasible
values for each one of the \( k \) variables and run the \( 2^k \) different experiments. This basic method is usually used as screening experiment, and it works well for binary variables or for the modelling of simple responses. If \( k \) is too high, a factorial design requires too many experiments, and one can run only a fraction of these \( 2^k \) experiments. Random or quasi-random techniques are also used in DoE, especially in the case for which the response in the design space is totally unknown a-priori. These techniques can identify the optimum area or give an indication on its direction. This is made possible by computing the surrogate model of the process response and finding the optimum set of variables minimizing this function.

Two types of surrogate model have to be distinguished. The case where new experiments can be run after the first set is called online approach [18]. In the offline approach, the database cannot be updated by new experiments in a potentially optimum zone of the design space. A higher number of points is then required to build a more sophisticated surrogate model, which can be obtained by data-fitting for instance (\textit{e.g.} polynomial, sinusoidal, exponential, or the sum of them). Some points can be excluded of the data to refine the fitting in the target area. An optimization algorithm is then run to determine the optimum variable values minimizing the surrogate model (last block in Fig. 3–6).

The online approach allows local updates of the surrogate model by defining an infill zone where an improvement is expected. New experiments in this zone will add new data points. The loop of Fig. 3–6 can be covered, as many times as necessary. Several methods can be used to determine the interesting areas. One of them, the kriging method, computes a predictor of the response surface and the possible error prediction. New data points are searched only where the predictor is minimized and where the error is maximized. The response surface is explored both in the predicted optimal area - to improve the local optimum - and in poorly explored area - increasing the chance of discovering the unknown
global optima. If the response of the system is supposed to be quasi-convex, a move
limit strategy, which updates the design variables boundaries, could be more efficient than
kriging.

In summary, it is important to keep in mind that there is no universal RSM-based
optimization method, and an adapted DoE has to be found for each new problem.

The focus in the studied case is the exploration of the response surfaces of the vir-
tual soil for three soil properties \((c, \varphi, A)\), measured from the simulations of two different
experiments (angle of repose and direct shear test). The design variables to be optimized
have been identified in the previous section as \((r_0, \mu_s, \mu_r)\). Since the methodology has to
be usable to model any type of soil, the target values for the geotechnical properties of the
virtual soil are unknown at this point. For this particular reason, an acceptable approxi-
mation of the properties response surfaces in the whole design space is required. Moreover,
EDEM uses a graphical user interface (GUI) and does not allow code programming. Each
new simulation has thus to be set manually. It is impossible to run an automated optimiza-
tion algorithm (no loop possible in Fig. 3–6). Thus, an extension of the full factorial DoE
is chosen. Each design variable will cover a range of different values, and combinations
of these values will be tested, as detailed in the section describing the construction of the
surrogate models (cf. Sec. 3.4.4).

3.4.2 Simulation of the direct shear test

To simulate the direct shear test described in Sec. 3.2, only a slice of the three-part
box illustrated in Fig. 3–1 was modelled, to minimize the number of particles involved in
the simulations. Since a high number of simulations were required for a satisfactory design
space exploration, the simulations had to be short. The design of experiment used was the
following. The dependence on \(\mu_s\) and \(\mu_r\) was investigated first by setting \(r_0 = r_0^* = 50\mu m\)
and exploring each \((\mu_s, \mu_r)\) couple in:
which makes 35 cases. Then, the effect of $r_0$ was studied by testing different mean particle radii for fixed couples $(\mu_s, \mu_r)$:

$$r_0 \in \{50\mu m; 80\mu m; 200\mu m; 500\mu m\}$$

In the following, $r_0^*$ is defined as the default mean particle radius, equal to 50$\mu m$. The virtual experimental setup used with the $r_0^*$ mean particle radius soil (the setup used for the coarser or finer soils was a simple homothety of the setup used for $r_0 = r_0^* = 50\mu m$) was modelled as follows. The soil consisted in 1,865 spherical particles with a size distribution parameter $\delta = 3$ (radii randomly distributed between 25$\mu m$ and 75$\mu m$). As shown in Fig. 3–7, the particles were compressed in a box made of a lower frame (in black), an upper frames (one part on each side, in dark blue) and a massless rectangular particle on top (dark red). This soil box was 10mm wide (distance between left and right walls) and 1.5mm deep (thickness of the slice), with periodic boundaries to simulate an infinitely deep box (Y direction in Fig. 3–7). The soil accumulated under the top cover was approximately 8mm high.

Figure 3–7: The virtual experimental setup used in the direct shear test simulations (for $r_0 = r_0^* = 50\mu m$ and at time $t = 0s$ here).
As explained in Sec. 3.2, the soil must be compressed at various pressures and one frame should slide horizontally while the shear reaction of the soil on the other frame should be recorded. An external vertical force was applied on the top particle through EDEM API to compress the soil. For each design variable set, four simulations were run, with forces of 0.75N, 1.125N, 1.5N and 2.25N, corresponding to pressures of 50kPa, 75kPa, 100kPa and 150kPa ($p = F/S = F/(0.01m * 0.0015m)$). Once the soil was compressed ($t = 0.05s$), a horizontal motion of 3mm/s was applied to the lower frame. The simulation was stopped at $t = 0.5s$. For each pressure $p$, the horizontal component of the reaction force of the soil on the lower frame was extracted (cf. Fig. 3–8); the mean value of the force on the plateau occurring at shear failure ($F_{\text{max}}$) was computed. The maximum shear stress was then computed as $\tau_{\text{max}} = F_{\text{max}}/S$, and cohesion and internal angle of friction were computed from the linear regression of $\tau_{\text{max}}$ as a function of $p$ (cf. Fig. 3–9).

![Figure 3–8: Shear force as a function of time extracted from direct shear test simulation for various pressure, for ($r_0 = r_0^*, \mu_s = 0.3, \mu_r = 0$).]
Figure 3–9: Linear regression of $\tau_{max}$ as a function of the pressure from direct shear test simulations, for ($r_0 = r_0^*, \mu_s = 0.3, \mu_r = 0$). The resulting cohesion and internal angle of friction are: $c = 3.02\,\text{kPa}$ and $\varphi = \arctan(0.590) \times \frac{180}{\pi} = 30.6^\circ$ (correlation: 0.999).

The simulation run time on a 3.20GHz 8-CPU workstation with 8GB of RAM was approximately 2 hours for a 0.5 second simulation with a given pressure $p$ and a given set of design variables. As the test has to be run for 4 different pressures per set of design variable, getting the soil properties $c$ and $\varphi$ at a given point of the design space required 8 hours of computational time. The timestep used was $\delta t = 8 \cdot 10^{-7}\,\text{s}$. The complete results of direct shear test simulations are given in Appendix E.

3.4.3 Simulation of the angle of repose experiment

In a paper published in 2009, Shunying Ji and Hayley Shen modelled angle of repose simulations using the discrete element method in 2 dimensions and investigated the effect of electrostatic charge on the one hand, and of static friction coefficient, within the linear spring contact model context, on the other hand, under Earth and Moon gravities [12]. Their simulation setup consists of a vertical tube filled with 698 circular particles or 349
paired particles (made of 2 attached circles), resting on a horizontal surface, as shown in Fig. 3–10.

Figure 3–10: The formation process of a two-dimensional angle of repose simulation: (a) the virtual soil rests in the tube, (b and c) the tube is lifted slowly, and (d) the simulation ends when the soil reaches a static state [12].

The angle of repose simulation was a 3D extension of S. Ji and H. Shen 2D simulations, and similar trends for angle of repose as a function of static friction coefficient were found, as detailed in Appendix G. The design of experiment used was comparable to the one used for the direct shear test, described in the previous section. The experimental setup for the simulations with a mean particle radius \( r_{0}^* \) consisted in a horizontal plate on which a 1.5mm diameter tube rested vertically. 3000 particles were generated inside the tube at the beginning of the simulation. The tube was lifted upward with a speed of 5mm/s, once the particles had settled down, which occurred one twentieth of a second after the particles generation. Approximately 0.2s after the beginning of the tube translation, the tube walls lost contact with the particle pile. The simulation was then run for another half a second in order to make sure that the pile reached its static state (cf. Fig. 3–11). Simulations involving larger particles were simple homotheties of this experimental setup.
$t = 0s$: the particles are generated in the tube.

(b) $t \approx 0.06s$: the particles repose on the steel plate.

c) $t \approx 0.2s$: the tube is lifted up.

(d) $t \approx 0.7s$: the static state is reached.

Figure 3–11: The 4 steps of a 3D angle of repose simulations.

The systematic method used to measure the angle of repose for a pile of particles works as follows (the complete MATLAB function is given in Appendix F). First, the cartesian coordinates of the center of the particles were extracted at the end of the simulation into a three-column matrix $M_1$. Then a plane $P$ passing by the axis of the tube, and hence intersecting the pile by its central axis, was considered. Any particle whose distance from $P$ was greater than $r_{max}$ was discarded from $M_1$, as well as any particle closer than $3r_{max}$ from the axis of the pile, $r_{max}$ being the maximum particle radius. Particles in contact with the ground plate were discarded too, as they might have rolled out of the main pile. Then, the remaining particles of the matrix $M_1$ were orthogonally projected on $P$. Their coordinates in the plane $P$ were stored in a new two-column matrix $M_2$. These two columns represent the abscissa, or distance from axis, and altitude of the projection of the particles. Finally, the $X$-axis of the plane $P$ was divided into intervals of $1.7r_{max}$ in length. At
each interval, the point with the highest altitude was the only one kept in $M_2$, the other ones being discarded. The points of $M_2$ having positive and negative abscissa were then considered separately, to compute the angle of repose on each side of the pile by a first degree polynomial fit. The arctangent of its first order term coefficient gives the angle of repose. Fig. 3–12 plots the angle of repose for various friction coefficients with $r_0 = r_0^*$. 

![Figure 3–12: Angle of repose of the virtual soil as a function of the static and rolling friction coefficients $\mu_s$ and $\mu_r$, for a mean particle radius $r_0^*$.](image)

This operation was repeated on 12 different planes uniformly distributed, giving 24 angles of repose. The final angle of repose of the pile was given by the mean of these 24 values. The repeatability of this angle of repose measure was investigated, concluding in an observational error of 7% (cf. Appendix D).

The simulation run time was approximately 2.5 hours for a 0.7 second simulation with a given set of design variables. The timestep used was $\delta t = 2.7 \cdot 10^{-7}s$, or 20% of the Rayleigh time step, given in Eq. [C-11]. The complete results of angle or repose simulations are given in Appendix E.
3.4.4 Building surrogate models for the virtual soil properties

Now that the design variables to calibrate have been identified and the virtual soil properties have been measured for various sets of design variables, the surrogate models need to be built from these data to express $c$, $\varphi$ and $A$ as a function of $r_0$, $\mu_s$ and $\mu_r$. The design variables vector is defined as

$$x = \begin{bmatrix} \mu_s \\ \mu_r \\ r_0 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$ 

The design of experiment focused first on exploring various values for $\mu_s$ and $\mu_r$ for $r_0$ fixed to $r_0^*$, and then investigating the effect of $r_0$ for several couples $(\mu_s$, $\mu_r)$. As a consequence, each surrogate model of a soil properties $i$ (which can be $c$, $\varphi$ or $A$) was built as a product of two independent functions:

- $f_i(\mu_s, \mu_r)$ describing the response surface for $r_0 = r_0^*$.
- $g_i(r_0)$, a dimensionless function of $r_0$ describing the dependence on the mean particle radius.

For instance, the surrogate model of the cohesion can be expressed as:

$$c(r_0, \mu_s, \mu_r) = f_c(\mu_s, \mu_r) \cdot g_c(r_0) \quad (3.3)$$

$f_i$ was built as the second-order fit of the data points collected for $r_0 = r_0^*$, since this fitting technique is simple and can approximate quite well curved patterns, such as the ones shown in Fig. 3–12. Hence, the functions $f_i$ are expressed as second-order polynomials:

$$f_i(\mu_s, \mu_r) = \sum_{j+k\leq2} \beta_{jk} \mu_s^j \mu_r^k \quad (3.4)$$

in which the $\beta_{jk}$ are the coefficients of the polynomial. The construction of the second order fit of the internal angle of friction is detailed as an example. The goal here is then to
find the coefficients $\beta_{jk}$ which minimize the errors between the values $f_\varphi(\mu_s^*, \mu_r^*)$ predicted by the surrogate model and the measures $\varphi^*(\mu_s^*, \mu_r^*)$ for all couples of parameters $(\mu_s^*, \mu_r^*)$ for which the soil property has been measured, noted as the set $\Omega$. To compute these coefficients, Myers and Montgomery’s method was used [22]. It uses a least squares based method, and finds the $\beta_{jk}$ minimizing the sum:

$$
\sum_{\Omega} \left[ \varphi^*(\mu_s^*, \mu_r^*) - \beta_{00} + \beta_{10}\mu_s^* + \beta_{12}\mu_r^* + \beta_{01}\mu_r^* + \beta_{02}(\mu_r^*)^2 \right]^2
$$

(3.5)

First, a table listing the measured internal angle of friction as a function of $\mu_s$ and $\mu_r$ is built (cf. Tab. 3–4).

Table 3–4: Measure of the virtual soil internal angle of friction from direct shear test simulations, with mean particle size $r_0^*$.

<table>
<thead>
<tr>
<th>Static friction coeff.</th>
<th>Rolling friction coeff.</th>
<th>Internal angle of friction $\varphi(\mu_s, \mu_r)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_s$</td>
<td>$\mu_r$</td>
<td>17.26</td>
</tr>
<tr>
<td>0.1</td>
<td>0</td>
<td>17.26</td>
</tr>
<tr>
<td>0.1</td>
<td>0.05</td>
<td>17.26</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>17.26</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>28.88</td>
</tr>
<tr>
<td>0.3</td>
<td>0.05</td>
<td>28.88</td>
</tr>
<tr>
<td>0.3</td>
<td>0.1</td>
<td>28.88</td>
</tr>
<tr>
<td>0.5</td>
<td>0</td>
<td>34.92</td>
</tr>
<tr>
<td>0.5</td>
<td>0.05</td>
<td>34.92</td>
</tr>
<tr>
<td>0.5</td>
<td>0.1</td>
<td>34.92</td>
</tr>
</tbody>
</table>

Then, an $X$ matrix and a $y$ vector are built from Tab. 3–4 as follows. The $i^{th}$ line of $X$ contains the values $[1, \mu_s, \mu_r, \mu_s^2, \mu_r^2, \mu_s\mu_r]$ of the $i^{th}$ line of Tab. 3–4, while the $i^{th}$
component of vector \( y \) is \( \varphi(\mu_s, \mu_r) \) (cf. Eq.[3.6]).

\[
X = \begin{bmatrix}
1 & 0 & 0.1 & 0 & 0.01 & 0 & 0 & 17.26 \\
1 & 0.1 & 0.05 & 0.01 & 0.0025 & 0.005 & 14.14 \\
1 & 0.1 & 0.1 & 0.01 & 0.01 & 0.01 & 7.16 \\
1 & 0.3 & 0 & 0.09 & 0 & 0 & 28.88 \\
1 & 0.3 & 0.05 & 0.09 & 0.0025 & 0.015 & 28.02 \\
1 & 0.3 & 0.1 & 0.09 & 0.01 & 0.03 & 12.87 \\
1 & 0.3 & 0.2 & 0.09 & 0.04 & 0.06 & 34.92 \\
1 & 0.5 & 0 & 0.25 & 0 & 0 & 33.47 \\
1 & 0.5 & 0.05 & 0.25 & 0.0025 & 0.025 & 30.33 \\
1 & 0.5 & 0.1 & 0.25 & 0.01 & 0.05 & 21.53
\end{bmatrix}
\]

\[
y = \begin{bmatrix}
17.26 \\
14.14 \\
7.16 \\
28.88 \\
28.02 \\
12.87 \\
34.92 \\
33.47 \\
30.33 \\
21.53
\end{bmatrix}
\]  \hspace{1cm} (3.6)

The unknown coefficients \( \beta_{j,k} \) are written as the vector:

\[
b = \begin{bmatrix}
\beta_{0,0} \\
\beta_{1,0} \\
\beta_{0,1} \\
\beta_{2,0} \\
\beta_{0,2} \\
\beta_{1,1}
\end{bmatrix}
\]  \hspace{1cm} (3.7)

The minimization of the sum expressed in Eq. [3.5] can then be written:

\[
\min_b \| y - Xb \|^2
\]  \hspace{1cm} (3.8)

If \( X \) was a square nonsingular matrix, the coefficients could be computed as: \( b = X^{-1}y \). But the problem is over-determined, and thus \( X \) is not square. To compute \( b \),

54
Myers and Montgomery use the left Moore-Penrose generalized inverse (LMPGI) of the matrix $X$: $(X'X)^{-1}X'$. The $\beta_{j,k}$ coefficients are then given by:

$$b = (X'X)^{-1}X'y = \begin{bmatrix} 7.127 \\ 109.3 \\ -72.95 \\ -109.1 \\ -258.5 \\ 121.5 \end{bmatrix} \tag{3.9}$$

The functions $f_i$ computed for each soil property are given here:

$$f_A(\mu_s, \mu_r) = 5.05 + 95.55\mu_r + 36.69\mu_s - 181.1\mu_r^2 - 18.98\mu_s^2 + 42.09\mu_s\mu_r$$

$$f_c(\mu_s, \mu_r) = 4.17 + 39.30\mu_r - 18.04\mu_s + 30.14\mu_r^2 + 2.38\mu_s^2 + 52.55\mu_s\mu_r \tag{3.10}$$

$$f_\phi(\mu_s, \mu_r) = 14.61 - 76.10\mu_r + 57.15\mu_s - 25.30\mu_r^2 - 25.57\mu_s^2 + 4\mu_s\mu_r$$

From this surrogate model, any property $i$ of a virtual soil characterized by any $(\mu_s, \mu_r)$ values and with a mean particle size $r_0^*$ can now be estimated via polynomial $f_i$. For instance, the internal angle of friction of a soil with $\mu_s = 0.4$ and $\mu_r = 0.07$ is given by:

$$f_\phi(\mu_s = 0.4, \mu_r = 0.07) = 28.62.$$

To take the variation with respect to $r_0$ into account, $f_i$ is multiplied by the dimensionless function $g_i$. For the angle of repose, it was found that the bigger the particles, the smaller the angle of repose. On a semi-log scale, a linear relationship was found, as shown in Fig. 3–13. For $\mu_s = 0.5$ and $\mu_r = 0.05$, the linear regression gave $A(r_0) = A(r_0^*) \cdot [1 - 0.117\ln r_0/r_0^*]$ (correlation coefficient 0.992). For $\mu_s = 0.9$ and $\mu_r = 0$, the linear regression gave $A(r_0) = A(r_0^*) \cdot [1 - 0.113\ln r_0/r_0^*]$ (correlation coefficient 0.988). Consequently, $g_A$ was set as: $g_A(r_0) = 1 - 0.115\ln r_0/r_0^*$, or in other words, the surrogate
model giving the angle of repose as a function of the design vector \( \mathbf{x} \) is:

\[
A(\mathbf{x}) = f_A(\mu_s, \mu_r) \cdot \left[ 1 - 0.115 \ln \frac{r_0}{r_0^*} \right].
\]  

(3.11)

Such linear dependence on \( r_0 \) on a semi-log scale could not be identified for \( c \) and \( \varphi \). \( g_c \) and \( g_\varphi \) were thus computed as second-order polynomial fits instead:

\[
c(\mathbf{x}) = f_c(\mu_s, \mu_r) \cdot \left[ 0.27 + \frac{r_0}{r_0^*} - 0.27 \left( \frac{r_0}{r_0^*} \right)^2 \right], \quad \varphi(\mathbf{x}) = f_\varphi(\mu_s, \mu_r) \cdot \left[ 0.71 + 0.44 \frac{r_0}{r_0^*} - 0.15 \left( \frac{r_0}{r_0^*} \right)^2 \right].
\]  

(3.12)

3.5 Search for optimum parameters via the Newton-Gauss algorithm

As explained in Sec. 3.1, the objective of the DEM parameter calibration is to bring the virtual soil properties \((c, \varphi \text{ and } A)\) as close as possible to the ones measured on the real soil to be modelled \((c_0, \varphi_0 \text{ and } A_0)\). In the previous section, surrogate models were built to describe the response surface of the virtual soil properties as a function of the parameters to be optimized: \(c(\mathbf{x})\), \(\varphi(\mathbf{x})\) and \(A(\mathbf{x})\). The problem is that the set of parameters for which the target cohesion \(c_0\) is reached might not give satisfactory \(A\) and \(\varphi\), or vice versa. Indeed,
an optimum vector $\mathbf{x}_0$ that would minimize three different and conflicting objectives was sought. This optimization problem is thus called a *multiple-objective optimization* (MOO) problem. First, the different methods used to solve MOO problems will be described in the next section. One of them, the Newton-Gauss algorithm, will be chosen and detailed in Sec. 3.5.2. Finally, it will be applied to the studied case in Sec. 3.5.3.

### 3.5.1 Basics of multi-objective optimization

An optimization problem can be defined as the minimization (or maximization) of objective functions by changing their design variables within certain constraint limits, called the design space. In the following, $\mathbf{x} = [x_1, x_2, \ldots x_n]^T$ represents an $n$-dimensional design vector and $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots \phi_m(\mathbf{x})]^T$ the $m$ objective functions depending on $\mathbf{x}$ to be optimized. Th $\varphi_i$ are assumed to be positive on the design space.

In the case $m = 1$, the objective vector is reduced to a single objective function $\phi(\mathbf{x})$ (cf. Fig. 3–14). The goal is then to find the global minimum of $\phi$ in the design space. One can distinguish two different classes of methods: gradient-based methods, involving steepest-ascent algorithms on convex functions for example, or probabilistic methods involving statistical or random search (for more complicated or unknown functions with several local minima) [15].

![Figure 3–14: Graphic representation of a single-objective optimization problem, which goal is to find the design vector $\mathbf{x}_0$ minimizing the objective function $\phi$ (here, $\phi$ is plotted as a function of a unique design variable).](image-url)
When $m > 1$, several objective functions should be minimized, although they might be conflicting, i.e. the optimum design vector $x_i$ minimizing $\phi_i$ can be different than $x_j$ minimizing $\phi_j$. The problem becomes a multi-objective optimization problem. Let us consider an MOO problem involving 2 conflicting objective functions $\phi_1$ and $\phi_2$ for instance ($m = 2$). If the image of the design space is plotted in the objective space (cf. Fig. 3–15), the design space can be separated in two subsets, $P_1$ (the red line in Fig. 3–15) and $P_2$ (the green area). To sort the design space in two subsets, one needs the following definition:

**Definition** Let $x$ and $y$ be two elements of the design space. $x$ dominates $y$ if and only if:

- $x$ is no worse than $y$ in all objectives.
- $x$ is strictly better than $y$ in at least one objective.

The subset $P_1$ and $P_2$ are then defined with the following proposition:

**Proposition 3.5.1** Any solutions of $P_1$ must be non-dominated with respect to each other. Any solution of $P_2$ is dominated by at least one solution of $P_1$.

With this proposition, any element $y \in P_2$ is not optimal as a solution $x$ dominating $y$ can be found in $P_1$. On the contrary, two solutions $x_1$ and $x_2$ in $P_1$ are equally optimal: if one of them is strictly better in one objective $\phi_i$, it is strictly worse in at least one other objective $\phi_j$. Such two solutions are called Pareto-optimal solutions, and without higher level information on the relative importance of the objectives, they are equally important.

The set $P_1$ is called the *Pareto front*. It is a hyper-surface of the design space.

Deb sorted the different methods used for MOO problems in three classes [8]:

- *a-priori methods*: The relative importance (weight) of each objective is chosen by the user before the optimization, and a unique solution minimizing the weighted sum of the objective functions is obtained. These methods are the least computationally
expensive, however they find only one point of the Pareto front, function of the weight configuration.

- **interactive methods**: The user interacts continuously with the optimization algorithm, leading to a trade-off solution.

- **a-posteriori** methods: A set of solutions describing the Pareto front is computed, as close to the Pareto front and as diverse as possible. Then, the user chooses one of these solutions, based on posterior informations and on the analysis of the approximated Pareto front obtained. These methods are more complex and computationally expensive. Examples: adaptive weighted sum, normal constraint, normal boundary intersection, non dominated sorting genetic algorithm.

The soil calibration is an MOO problem: in most cases, it will not be possible to reach exactly all the target properties \((c_0, \varphi_0 \text{ and } A_0)\) of a real soil. To solve this optimization problem, the Newton-Gauss algorithm, an *a-priori* method, will be used, for its simplicity and because no information about the relative importance of the different properties \(c, \varphi\) and \(A\) on the soil modelling quality have been found in the literature. It is then assumed
that all properties have equal importance in the calibration process. However, the Newton-Gauss method includes a weighting matrix. Changing this matrix will give a different point of the Pareto front by computing the optimum solution for different properties relative importance.

3.5.2 Presentation of the Newton-Gauss optimization algorithm

The Newton-Gauss algorithm used to solve the optimization problem is presented here [1].

Let us consider an \( n \)-dimensional design vector \( \mathbf{x} = [x_1, x_2, \ldots, x_n]^T \) to optimize to verify the following system of nonlinear equations:

\[
\begin{bmatrix}
    \phi_1(\mathbf{x}) \\
    \phi_2(\mathbf{x}) \\
    \vdots \\
    \phi_m(\mathbf{x})
\end{bmatrix} = \mathbf{0}.
\]  

(3.13)

Let us assume that this system is overdetermined, which means \( m > n \). The Newton-Gauss method will look for a least-square error to minimize the overall error of the nonlinear system. It will then minimize the function \( f \) defined as

\[
f(\mathbf{x}) = \frac{1}{2} \phi^T \mathbf{W} \phi
\]

(3.14)

where \( \mathbf{W} \) is a \( m \)-by-\( m \) positive-definite weighting matrix. If the same weight is given to each \( \phi_i \) function for instance, the weighting matrix would be

\[
\mathbf{W} = \mathbf{I}_m.
\]

(3.15)

The Newton-Gauss method works as follows: first, an initial guess \( \mathbf{x}^0 \) is specified. Then, as long as the convergence criterion is not reached, a increment \( \Delta \mathbf{x}^k \) multiplied by
a damping factor $\alpha$ is added to the solution vector $x^k$ iteratively:

$$x^{k+1} = x^k + \alpha \Delta x^k$$  \hspace{1cm} (3.16)

The damping factor $\alpha$ is comprised between 0 and 1 and avoids oscillations of the objective function $f(x)$ which could lead to divergence. $\Phi(x)$ is defined as the $m$-by-$n$ Jacobian of the vector function $\phi(x)$:

$$\Phi(x) = \frac{\partial \phi(x)}{\partial x}$$  \hspace{1cm} (3.17)

The increment $\Delta x^k$ is computed as follows:

$$\Delta x^k = -(\Phi^T W \Phi)^{-1} \Phi^T W \phi$$  \hspace{1cm} (3.18)

The convergence criterion is reached when the increment $\Delta x^k$ is small enough, or in other words when

$$\left\| \Delta x^k \right\| < \epsilon$$  \hspace{1cm} (3.19)

where $\epsilon$ is a specified positive tolerance.

When it converges, this procedure does so to a local minimum of the function $f$. The global minimum of $f$ in a specific design space can be obtained by specifying various initial guesses $x^0$ and, if it leads to different local optima, by selecting the optimum vector for which $f$ is minimum. Appendix K demonstrates the efficiency of the Newton-Gauss algorithm on a simple analytical example.

### 3.5.3 Results of the calibration problem for lunar regolith and silica Barco sand

For the soil calibration case, three objective functions to be minimized are defined as the normalized distance between virtual and target soil properties:

$$\phi_1(x) = \left| \frac{A(x) - A_0}{A_0} \right|, \quad \phi_2(x) = \left| \frac{c(x) - c_0}{c_0} \right|, \quad \phi_3(x) = \left| \frac{\varphi(x) - \varphi_0}{\varphi_0} \right|$$  \hspace{1cm} (3.20)
in which \( \mathbf{x} = (\mu_s, \mu_r, r_0) \) describes the design vector and \( A_0, c_0 \) and \( \varphi_0 \) are the target soil properties (silica sand or lunar regolith properties given in Tab. 3–1 for instance). Moreover, a fourth objective function \( \phi_4 = \phi_4(r_0) \), positive and decreasing towards zero when \( r_0 \) increases, is defined to illustrate the preference for bigger particles. Indeed, the bigger the mean particle radius, the fewer the number of particles needed to simulate the virtual soil in the future simulations, and thus the shorter the simulation run time. The following function was used:

\[
\phi_4(\mathbf{x}) = 10 \left( \frac{\pi}{2} - \arctan(r_0 - 5) \right)
\]

with \( r_0 \) expressed in microns and dimensionless. Fig. 3–16 plots \( \phi_4(r_0) \) for \( r_0 \) from 10\( \mu m \) to 500\( \mu m \).

![Figure 3–16: Objective function \( \phi_4 \) as a function of the mean particle radius \( r_0 \), positive and decreasing towards zero, used in the calibration optimization process to express the preference for large particles.](image)

The calibration to reach silica sand properties worked well. The algorithm converged in 19 iterations with a convergence criterion \( \epsilon = 0.01 \), a damping factor \( \alpha = 1 \), an initial guess
\( x^0 = (\mu_s = 0.5, \mu_r = 0.05, r_0 = 50\mu m) \) and equal weights for all 4 objective functions, leading to predicted virtual soil properties close to the real ones, as shown in Tab. 3–5. The optimum design vector computed was: \( x_0 = (\mu_s = 0.609, \mu_r = 0.0811, r_0 = 70.9\mu m) \).

To make sure that this optimum design vector \( x_0 \) was a global optimum and not just a local one, the algorithm was run with several other initial guesses. All of them led to the vector \( x_0 \), with the exception of some initial guesses with a very high static friction coefficients, leading to non-feasible solutions (outside the design space). For instance, \( x^0 = (\mu_s = 1.5, \mu_r = 0.15, r_0 = 100\mu m) \) led to the solution \( x_0 = (\mu_s = 1.78, \mu_r = 0.149, r_0 = 111\mu m) \), rejected because \( \mu_s = 1.78 \) is out of the limits.

For the lunar regolith, several cases were considered. The lunar soil does not have homogeneous properties on the whole surface of the Moon, as confirmed by the literature (cf. Tab. 3–1). Two possibilities for \( c \) (\( c_0 = 0.1kPa \) and \( c_0 = 1kPa \)), and three possibilities for \( \varphi \) (\( \varphi = 30^\circ \), \( \varphi = 40^\circ \) and \( \varphi = 50^\circ \)) are considered to model various types of lunar soils.

In some cases, the algorithm could not converge unless the damping factor was reduced to \( \alpha = 0.001 \). Moreover, the regolith angle of repose is very high (\( A_0 = 50^\circ \)) and was barely reached in the angle of repose simulations, as shown in Fig. 3–12. For that reason, a modified weighting matrix \( W_m \) was used, decreasing the importance of the angle of repose

<table>
<thead>
<tr>
<th>Target property value</th>
<th>Predicted virtual property after calibration</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_0 = 24Pa )</td>
<td>( c = 24Pa )</td>
</tr>
<tr>
<td>( \varphi_0 = 26.6^\circ )</td>
<td>( \varphi = 29.5^\circ )</td>
</tr>
<tr>
<td>( A_0 = 30^\circ )</td>
<td>( A = 27.9^\circ )</td>
</tr>
</tbody>
</table>
Using $W_m$ instead of $W = 0.25I_4$ emphasizes more accurate $c$ and $\varphi$ values, with the angle of repose farther from the target value $A_0$. Tab. 3–6 sums up the calibrated parameters for various lunar regolith scenarios, starting from the initial guess $x^0 = (\mu_s = 0.5, \mu_r = 0.05, r_0 = 50\mu m)$. The same results were obtained with different initial guesses.

Table 3–6: Calibration results for different lunar regolith scenarios, with damping factor $\alpha = 0.1$, convergence criterion $\epsilon = 0.01$ and initial guess $x^0 = (\mu_s = 0.5, \mu_r = 0.05, r_0 = 50\mu m)$.

<table>
<thead>
<tr>
<th>Target property values $(A_0, c_0, \varphi_0)$</th>
<th>$W$</th>
<th>Predicted virtual property after calibration</th>
<th>Design vector $x_0 = (\mu_s, \mu_r, r_0 [\mu m])$</th>
<th>Iterations to converge</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(50\circ, 0.1kPa, 30\circ)$</td>
<td>$0.25I_4$</td>
<td>(37.0\circ, 101Pa, 32.4\circ)</td>
<td>(1.32, 0.139, 61.6)</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>$W_m$</td>
<td>(27.9\circ, 89.8Pa, 31.4\circ)</td>
<td>(0.645, 0.0871, 97.1)</td>
<td>37</td>
</tr>
<tr>
<td>$(50\circ, 1kPa, 30\circ)$</td>
<td>$0.25I_4$</td>
<td>(38.4\circ, 101kPa, 32.0\circ)</td>
<td>(1.29, 0.152, 61.9)</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>$W_m$</td>
<td>(30.0\circ, 100kPa, 31.2\circ)</td>
<td>(0.682, 0.108, 99.3)</td>
<td>66</td>
</tr>
<tr>
<td>$(50\circ, 0.1kPa, 40\circ)$</td>
<td>$0.25I_4$</td>
<td>(34.8\circ, 99.7Pa, 35.7\circ)</td>
<td>(1.20, 0.134, 104.6)</td>
<td>1448 $(\alpha = 0.005)$</td>
</tr>
<tr>
<td></td>
<td>$W_m$</td>
<td>(32.1\circ, 100Pa, 39.7\circ)</td>
<td>(1.21, 0.134, 199)</td>
<td>1881 $(\alpha = 0.005)$</td>
</tr>
<tr>
<td>$(50\circ, 1kPa, 40\circ)$</td>
<td>$0.25I_4$</td>
<td>(35.5\circ, 100kPa, 35.2\circ)</td>
<td>(1.20, 0.145, 105.8)</td>
<td>1279 $(\alpha = 0.005)$</td>
</tr>
<tr>
<td></td>
<td>$W_m$</td>
<td>(32.5\circ, 100kPa, 39.5\circ)</td>
<td>(1.20, 0.141, 208)</td>
<td>1870 $(\alpha = 0.005)$</td>
</tr>
<tr>
<td>$(50\circ, 0.1kPa, 50\circ)$</td>
<td>$0.25I_4$</td>
<td>(32.5\circ, 98.8Pa, 39.1\circ)</td>
<td>(1.19, 0.132, 179)</td>
<td>3023 $(\alpha = 0.002)$</td>
</tr>
<tr>
<td></td>
<td>$W_m$</td>
<td>(30.4\circ, 66.2Pa, 41.2\circ)</td>
<td>(1.15, 0.130, 284)</td>
<td>1381 $(\alpha = 0.002)$</td>
</tr>
<tr>
<td>$(50\circ, 1kPa, 50\circ)$</td>
<td>$0.25I_4$</td>
<td>(33.2\circ, 997Pa, 38.6\circ)</td>
<td>(1.19, 0.141, 177)</td>
<td>5444 $(\alpha = 0.001)$</td>
</tr>
<tr>
<td></td>
<td>$W_m$</td>
<td>(31.0\circ, 972Pa, 40.8\circ)</td>
<td>(1.16, 0.138, 285)</td>
<td>1979 $(\alpha = 0.002)$</td>
</tr>
</tbody>
</table>

As shown in Tab. 3–6, the calibration algorithm could not reach high angle of repose and internal angle of friction. This is possibly due to the choice of single-sphere particle
prototype, leading to a low interlocking of particles. Unlike the silica sand which properties could be reached satisfactorily by the virtual material, the lunar regolith might require a finer and more complex modelling.

### 3.5.4 Validation of the silica sand calibration with simulations

In the previous section, the Newton-Gauss algorithm could determine an optimum set of parameters $x_0$ that would set the virtual soil properties near to the silica sand properties, as shown in Tab. 3–5. However, the predicted virtual soil properties are values predicted by the surrogate models built from a finite number of simulations, for various design vectors. To validate the calibration, angle of repose and direct shear test simulations were ran using the optimum design vector $x_0 = (\mu_s = 0.609, \mu_r = 0.0811, r_0 = 70.9\mu m)$ as for the soil parameters. The results are the following:

- **Angle of repose:** predicted by the surrogate models: 27.9°; measured: 30.01°.
- **Cohesion:** predicted by the surrogate models: 24Pa; measured: 364Pa.
- **Internal angle of friction:** predicted by the surrogate models: 29.5°; measured: 33.4°.

These results are close to the predicted values given by the surrogate models, with the exception of cohesion, which is more than one order of magnitude higher than predicted - 364Pa instead of 24Pa. Since the objectives of this MOO problem are conflicting, no solution can fully reach all objectives, and the optimum solution found is the one having the least overall error. The high cohesion of the virtual soil does not pose a significant problem though, since soils having a cohesion less than 1kPa are considered as almost cohesiveless. The maximum shear stress $\tau_{max}$ of such soils depends much more on $\varphi$ than on $c$. According to the Mohr-Coulomb law (cf. Eq. [2.1]), and if the soil is under a pressure $p = 10kPa$ corresponding to a mid-size rover, the maximum shear stress of the virtual soil
will be $\tau_{\text{max}} = 364 + 10.000 \tan 33.4^\circ \approx 6.96 kPa$, instead of $\tau_{\text{max}} = 24 + 10.000 \tan 33.4^\circ \approx 6.62 kPa$, which corresponds to a 5% increase.

To conclude, once calibrated to model the silica sand behavior, the virtual soil properties are set as close as possible to the real ones, within the soil modelling constraints. No other design variable values can decrease the overall error for the selected properties. The values for these properties are close enough to the real ones to assume a high fidelity in the soil behavior. However, the validity of wheel-soil simulations using this soil model has to be demonstrated, which is done in the next chapter. Moreover, one should keep in mind at all times that the soil modelling can be constantly perfected, by increasing the number of design variables or target properties for instance.

3.6 Summary

The calibration of the virtual soil parameters involved in the DEM contact model and characterizing the material properties and interaction coefficients is a crucial preliminary step for accurate wheel-soil simulations. A four step methodology, based on geotechnical experiments made on the real soil and repeated numerically, was proposed in this chapter to minimize the discrepancies between selected properties of the real and virtual soils - angle of repose, cohesion and internal angle of friction. A silica sand could successfully be modelled with realistic properties, paving the way for wheel-soil simulation experimental validation. However, the soil model has to be improved to reach the lunar regolith properties. Indeed, more design variables and target properties and a more complex particle shape need to be considered in order to model this very particular material accurately, although the methodology remains valid.
CHAPTER 4
Wheel-soil interaction simulations

This chapter first presents the two approaches used in the present study to model wheel-soil interactions in a DEM environment: a displacement-controlled approach in which the wheel is built as a geometry object and its motion is fully determined as a combination of a translation and a rotation, and a torque-controlled approach in which the wheel is considered as a particle made of an assembly of spheres, on which external forces are applied through EDEM’s API, and which motion ensues from the Newton’s equations of motion. The method used to parametrize and build the wheel in the simulation system and to extract simulation data is described for each case. Then, the validity of DEM-based simulations using both approaches is demonstrated by comparing simulated results to experimental and theoretical ones. Finally, the use of this simulation environment as an optimization tool is described. The existence of an optimum grouser configuration is demonstrated and the architecture of an automated optimization algorithm is proposed.

4.1 Two approaches for wheel-soil simulations using EDEM

In the previous chapter, the accurate modelling of granular materials, such as silica sand, was described. This was a preliminary step to wheel-soil interaction simulations. Now, the goal is to put a wheel on this virtual soil and make it roll to evaluate its performances.

CPU resources is the first limitation of DEM simulations. For instance, the runtime of a 1 second simulation involving 10,000 millimetric particles is approximately one hour with
the 8 core 3.2GHz workstation available in the laboratory. As a comparison, the single-wheel test bed used for experimental validation was filled with roughly one cubic meter of silica sand, corresponding to more than \(10^{10}\) grains. It is then obviously impossible to model the whole test bed in EDEM, as each experiment would take millions of hours to be reproduced numerically. To deal with this constraint, previous DEM-based wheel-soil simulations available in the literature were two-dimensional and involved particles larger than the real soil, as detailed in Sec. 2.3.

To prove the feasibility of 3D DEM-based wheel design optimization compatible with the available CPU resources, it was decided to limit the simulations to a thin slice of the wheel-soil system. The rigid soil bin containing the virtual soil was then modelled as an elongated rectangle, with a soil depth of 5 to 10 times the bin width, to guarantee that the boundary effect of the bottom of the bin was negligible (cf. Sec. 2.1.1). The sides of the bin were either solid or set as periodic boundaries to model an infinitely wide system. In the latter case, the bin width was set to at least a dozen particle diameters to avoid any interactions between a given particle and itself through the periodic bin sides. This longilineal system restricted the simulations to straight rolling in the plane \((O, X, Z)\), the behavior of the wheel in turns is left out in this study.

As explained in Sec. 2.2.2, any simulation system in EDEM consists in particles and geometries. The most intuitive way to model the wheel is to use a geometry, as EDEM allows the importation of CAD files. However, this CAD-wheel method has a major drawback. The motion of each geometry is completely defined by the user in the pre-processing step of the simulation and does not depend on the wheel-soil interactions computed during the simulation. Hence, this CAD-wheel approach is called the displacement-controlled approach. To bypass this limitation and to have a wheel motion driven by the Newton’s
equations of motion, another wheel modelling method will be used in a different set of sim-
ulations. The wheel will be made of an assembly of spheres and considered by EDEM as a
particle object instead of a geometry object, allowing a motion control based on wheel-soil
interactions. This will be called the \textit{torque-controlled} approach. These two wheel modelling
approaches, and their pros and cons, are described in details.

4.1.1 Displacement-controlled wheels

In this section, the simulations involving wheels modelled as geometries are described.
The wheel parametrization and construction is addressed first, the simulation outputs and
data extraction possibilities follow.

\textbf{Wheel construction}

Although it is relatively simple to import a geometry from a CAD model into the
EDEM simulation system, the geometry has to be positioned at the desired coordinates in
the CAD software before importing it in EDEM, as it cannot be moved or rotated once
imported. EDEM decomposes a CAD file into nodes, and the surfaces of the geometry
are defined as an assembly of triangles, each triangle being defined by a node triplet. This
forbids the use of curved shapes such as cylinders for instance.

Moreover, the motion of geometries in EDEM can only be defined through the graph-
ical user interface (GUI) in the pre-processing step by the user, and they can only be a
combination of linear translations and a linear rotations, with constant initial velocities and
accelerations - this is why it is called the \textit{displacement-controlled} approach. This means
that for every wheel shape change, a new CAD model has to be built, imported in EDEM,
and its motion reset through the GUI. These steps are very time consuming and needed
to be repeated a large number of times due to the nature of the present work. Design
optimization requires a high number of iterations to be efficient. An automated alternative
was found to this manual configuration process, using the unofficial and unsupported XML
input deck feature, through which the whole simulation system could be set in an XML file, including soil definition, particle positions, geometry nodes, triangles and motions. The “soil” part of the XML input deck is not worthwhile to be described here as it only consists in particle properties and positions. The focus is on the assembly of the “wheel” part of the XML file.

Before starting to build the geometry wheel per se, i.e. to compute the nodes’positions and node triplets, the wheel parametrization has to be detailed. 8 parameters were chosen to characterize the wheel, as shown in Fig. 4–1:

- $R$: wheel radius.
- $w$: wheel width.
- $N_G$: number of grousers.
- $h$: height of grousers.
- $\alpha$: angle of V-Shaped grousers.
- $\beta$: angle of the grouser front side with the radial direction.
- $w_b$: grouser width at its base.
- $w_t$: grouser width at its top.

Now, the position of each node can be computed. The nodes forming the first grouser at the top of the wheel are computed first, then the coordinates of the nodes forming the other grousers are computed by a simple rotation around the line parallel to the (OY) axis and passing by the center of the wheel. The XML code describing the wheel is simply composed of the list of nodes ($id$ and positions) and the list of triangles (node triplet), as shown in the following example:

```xml
<nodes>
  <node x="0.093389" y="0" z="0.16341" id="20501" />
  <node x="0.082129" y="0.005" z="0.16304" id="30436" />
  ...
</nodes>
<triangle id="355" node1="10336" node2="10136" node3="10236" />
```
Two parameters are not shown on this picture: the wheel radius $R$ and the number of grousers $N_G$.

Fig. 4–2 shows a close-up of the tread of a 16-grouser wheel on which the triangles forming the wheel structure appear clearly. The MATLAB function used to assemble the XML code from a list of parameters is given in Appendix H (4 other parameters, $h_s$, $\beta_s$, $w_{bs}$ and $w_{ts}$, describing the grouser sides, are added in this function to allow more complex grouser shapes).

**Motion control**

As previously mentioned, the wheel motion is completely defined *a-priori* by the user in displacement-controlled simulations, by specifying translations and rotations. However, EDEM has an unofficial and unsupported feature called *servo-control*, allowing to specify a constant reaction force on a geometry in each one of the three cartesian directions: $F_X$, $F_Y$ and $F_Z$. The given geometry will then move along each direction until the particles in the simulation domain exert the specified reaction force. $F_Z$ was the only one set to a non-zero value, equal to the supposed load applied on the wheel. At the beginning of the simulation, the wheel is formed just above the soil and the reaction force $F_Z$ is set to let
Figure 4–2: Close-up on a 16-grouser V-shaped groused wheel geometry showing the triangles forming the wheel.

it sink into the soil until the reaction $F_Z$ is reached, as shown in Fig. 4–3. This usually takes less than half a second.

Figure 4–3: A wheel resting on the soil in the displacement-controlled approach, before the rotation and translation start.

Then, a linear translation in the X direction at the velocity $v$ combined to a rotation around the line parallel to the Y axis and passing by the center of the wheel are applied. The rotational velocity $\omega$ is computed to reach the desired slip, thanks to Eq. [2.6].
Simulation outputs

For most simulation using geometry wheels in the displacement-controlled approach, the data extracted from the simulations were the following:

- X component of the total reaction force of the soil on the wheel (corresponding to the instant net traction).
- Z component of the total reaction force of the soil on the wheel.
- Y component of the total reaction torque of the soil on the wheel about its axis of rotation (corresponding to the opposite of the virtual instant torque that would be applied by the shaft of a motor to the wheel).
- wheel position and velocity, to get the wheel sinkage.

This displacement-controlled approach allows the modelling of wheels with complex grouser shapes. However, the user inputs the velocity and the slip a-priori as if the wheel was driven by an infinitely powerful arm capable of giving it any possible motion, and gets the resulting forces and torques from the simulation outputs. In reality, one controls a wheeled vehicle like a car or an electric rover by specifying a power input on the motor or a torque on the wheel. The position, velocity and acceleration of the wheel are then direct consequences of the interaction forces, through Newton’s equations of motion. This is what is sought in the torque-controlled approach.

4.1.2 Torque-controlled wheels

In order to apply customized efforts such as torque, weight or drawbar load to a wheel in an EDEM simulation, the wheel has to be modelled as a particle instead as a geometry. Indeed, EDEM’s API allows to apply external forces coded in C++ plug-ins to a given type of particle. EDEM particles can only be made of spheres (cf. Sec. 2.2.2), thus the wheels are built as assemblies of juxtaposed spheres, as in Fig. 2–11 by Nakashima et al.
Wheel construction

A MATLAB function, given in Appendix I, was written to build the XML code describing the positions of the particle spheres depending on the following parameters: radius and width of the wheel, number, height and thickness of the grousers, radius of the spheres forming the particle, and overlapping of neighboring spheres. Ideally, the spheres forming the wheel should be small and neighboring spheres should have a high overlap to increase the wheel shape accuracy and smoothness, but the more spheres involved in the particle, the longer the simulation runtime. Thus, super-millimetric spheres were chosen, with a distance between two contacting spheres of 1.3 sphere radius. Fig. 4–4 gives an example of such a particle wheel, and an example of sphere definition in the XML file built by the MATLAB function and describing the wheel follows:

```
  <sphere x="0.0090297" y="0" z="0.05121" contact_radius="0.002" id="123" name="sphere123" physical_radius="0.002"/>
```

Figure 4–4: An 18-grouser particle wheel made of an assembly of spheres, resting on the soil in a torque-controlled simulation.

One drawback of using EDEM’s API to include external plug-ins is that it roughly multiplies the simulation runtime by a factor 5. To keep reasonably short simulations, the
number of soil particles in simulations using the torque-controlled approach was thus lower than in the case of displacement-controlled ones.

Motion control

When using particle wheels, only the shell of the wheel tread was modelled (cf. Fig. 4–4). It was necessary to use wheels as wide as the soil bin, to guarantee that no soil particle would enter the wheel by the sides. As a consequence, the interactions between the soil and the sides of the wheels could not be studied. The sides of the soil bin were then set as periodic boundaries, simulating an infinitely wide system. To ensure that the wheel would stay vertical and roll straight without falling on the side, a rigid wheel guide of the exact width of the particle wheel was added (cf. Fig. 4–5). This wheel guide was associated to a material called teflon, for which all interaction coefficients were set to zero, to avoid any energy dissipation through friction between the wheel and the guide.

Figure 4–5: A simulation using the torque-controlled approach. The particle wheel is forced to remain vertical and to roll in the X direction thanks to the frictionless wheel guide (in transparent blue on this picture).

Several external plug-ins were coded to allow a variety of motion control. Their code is given in Appendix J. All of them can apply a vertical load $W$ to the wheel, representing the weight of the rover. Moreover, the PolyTorque plug-in applies on the wheel a second-order
polynomial torque $T$ function of the time. The torque is always applied half a second after the beginning of the simulation, to ensure that the wheel reached its static sinkage position before starting to roll. Once the wheel is subject to a given torque, it starts to roll due to the traction effort of its tread on the soil, or it stays immobile if the torque is not sufficient to make it move.

The *ServoSpeed* plug-in works as if the controller drives at a constant velocity $v_0$, by checking the velocity $v$ of the wheel at each time step and applying a given driving torque $T$ when $v < v_0$.

The *ServoTorque* plug-in is a bit more sophisticated as it models the behavior of an electric motor. The torque map of an electric motor can be modelled as follow: it delivers its maximum torque $T_{\text{max}}$ when its rotational speed $\omega$ is under a threshold $\omega_{\text{max}}$, and it delivers no torque when $\omega > \omega_{\text{max}}$, as shown in Fig. 4–6.

![Figure 4–6: The simple modelling of an electric motor torque map: $T = T_{\text{max}}$ when $\omega < \omega_{\text{max}}, T = 0$ otherwise.](image)

**Simulation outputs**

In addition to the forces and interactions that could be obtained from displacement-controlled simulations (such as translational velocities, rotational velocities, slip or sinkage), new simulation outputs are relevant in torque-controlled simulations: data related to the
energy consumption can be obtained. As the torque is now controlled and defined by the user, the energy consumption can be computed as the torque multiplied by the wheel rotational velocity. This information is crucial to compare the performances of different wheels.

4.2 Validation of the results

A methodology to calibrate the virtual soil was presented in Chap. 3, and two approaches for wheel modelling in Sec. 4.1. Now, the model is compared to other works. Several comparisons with various experimental, numerical and theoretical results are presented in the following sections to confirm the validity of the model.

4.2.1 Experimental validation using a reduced scale vehicle

In this section, the power consumption of a wheel rolling at a constant speed on an silica sand soil is measured in a simulation and compared to experimental results collected by Kaveh-Moghaddam [13]. In the experimental part, Kaveh-Moghaddam used the Wheely King remote controlled reduced scale rover, shown in Fig. 4–7. It was driven on a flat area at the constant speed of 0.3 m/s, and the power consumption was measured with a voltmeter and an ammeter connected to the rover battery. The rover weighs 2.4 kg, and the power consumption was measured for the rover only or with various additional loads from 0 to 3 kg. The Wheely King rover was mounted on compliant Mud Thrasher rubber wheels (cf. Fig. 4–7). These 135 mm-diameter and 70 mm-wide wheels have 18 V-shaped grousers, 5 mm height and 8 mm wide.

To replicate this experiment numerically, a few assumptions were made. First, the rubber wheels were considered rigid. Indeed, even when an external load of 30 N was added on the rover standing on a rigid surface, the wheel deformations were negligible. The simulation system was composed of a 4 mm-wide and 500 mm-long soil bin, filled with a 39 mm-thick layer of virtual soil composed of 25,388 soil particles (cf. Fig. 4–9 (top)). The
DEM parameters found in Sec. 3.5.3 for the calibration of the virtual silica sand were used. The particle radius was augmented to 700 microns to keep a reasonably low number of particles in the system. The Mud Thrasher wheel was modelled as a particulate wheel using the MATLAB function given in Appendix I with the real rubber wheel parameters:

- Wheel diameter: 135mm.
- Number of grousers: 18.
- Grouser height: 5mm.
- Grouser width: 8mm.
- Diameter of the spheres forming the particle wheel: 4mm.
- Thickness of the particle wheel: 4mm.

The particle wheel was then a 4mm-wide slice of the real rubber wheel (cf. Fig. 4–8). The V shape of the real wheel grousers was neglected in the virtual wheel model and it was assumed that the wheel-soil interactions were constant along the wheel width. Consequently, the sides of the virtual soil bin were set as periodic boundaries, meaning that the wheel was assumed to be infinitely wide. The simulated wheel motion was restrained by a frictionless wheel guide (cf. Fig. 4–5).
Figure 4–8: Comparison between the real Mud Thrasher wheel (left, ©HPI Racing) and the 4mm-wide particle wheel used in the simulations (right).

The ServoSpeed plug-in was used to apply external forces on the wheel. The applied vertical load was computed as the total weight of the rover (rover weight plus additional load) divided by the sum of the 4 wheel width (4 times 70mm), multiplied by the virtual wheel width (4mm) times the gravitational acceleration. For instance, the external vertical load $F_Z$ applied for the simulation of the rover without additional load would be:

$$F_Z = \frac{2.4[kg] + 0[kg]}{4 \times 0.07[m]} \cdot 0.04[m] \cdot 9.8[m.s^{-2}] = 0.3360[N].$$ (4.1)

After $t = 0.3s$, a torque $T_{max} = 0.03Nm$ was applied to the wheel, when the horizontal speed was lower than $0.3m/s$. It took 1.5s for the wheel to settle and roll to the end of the 500mm soil bin, for a simulation runtime of roughly 4 hours. Fig. 4–9 shows the simulation system at two different times.

The total energy consumption $E$ was then extracted from the simulation data. When plotted as a function of the time, a steady state, where $E$ grew linearly with time, was reached after $t = 0.5s$ as shown in Fig. 4–10, corresponding to the constant speed regime.
Figure 4–9: The Wheely King analog simulation system, at two different times.

A linear regression of this part was then computed, and the slope was taken as the instant power consumption of the wheel $P_c$. An instant power consumption per unit wheel width $P_u$ was found as the ratio between $P_c$ and the wheel width, equal to 4mm. Finally, the total rover power consumption was estimated as $P_u$ times the sum of the rover wheel width: $4 \cdot 70mm = 280mm$.

To check the validity of the “slice” wheel assumption according to which a 4mm-thick wheel in a periodically bounded box would behave like a full size one, one of the simulation was repeated in a soil bin three times wider, (12mm instead of 4mm, with a 12mm-thick wheel). The estimated rover power consumptions were found to be very similar: 6.81W for the 4mm-thick simulation, and 7.04W for the 12mm-thick one. This proves that a large 3D system such as a wide wheel can be simulated as a relatively thin slice, ignoring the side effects. It should be mentioned here that the sinkage of the wheels in these experiments was very low (a few millimeters), leading to negligible wheel-soil interactions on the sides of the wheels.
The results of the complete experimental and simulated power consumptions are given in Fig. 4–11. The real power consumption is higher than the simulated one, with a higher discrepancy at low loads. This difference between experiments and simulations decreases when a heavy external mass is attached on the rover (total load of more than 40N).

The reason of the high discrepancy at lower loads could be due to the power measurement method. The instant power measured on the rover in the experiments is the power delivered by the battery, whereas in the simulation, it is the power delivered by a virtual shaft connected to the wheel. The losses in the electric motor and the frictional losses in the mechanical transmission from the motor to the wheels could be the reason of higher experimental power consumption.

In order to prove the importance of the soil calibration step, the simulation with the external load $W = 38.22N$ was run again with a soil calibrated to fit the geotechnical
properties of the last lunar regolith scenario given in Tab. 3–6. As given in this table, the estimated properties of the virtual soil were then: $A = 31.0^\circ$, $c = 972 Pa$, $\varphi = 40.8^\circ$, and the DEM parameters were set as: $\mu_s = 1.16$ and $\mu_r = 0.138$. The rover consumption was found to be $4.49W$ instead of the $5.62W$ required on the silica sand. This lower power consumption might stem from the higher cohesion of the virtual regolith ($c = 972 Pa$ instead of $c = 24 Pa$ for the virtual silica sand), which reduces the particle movements in the soil and thus the energy losses due to friction. This is confirmed by the lower sinkage in the regolith than in the silica sand: $2.53mm$ in the regolith on average, instead of $4.84mm$ in the silica sand. Two interesting conclusions can be made on these observations. First, it highlights the dependence of the results on the soil calibration. A 25% difference in the power consumption can result with the use of a different soil configuration. Secondly, it validates the calibration methodology by showing that the simulation results are closer to the experimental ones, when using a virtual soil calibrated to fit the properties of the real soil.
4.2.2 Comparison to other DEM wheel-soil interaction simulations

As cited in Sec. 2.3, Nakashima et al. compared in 2D DEM wheel-soil interaction simulations the traction and the sinkage of a given wheel, subject to the same external efforts, on terrestrial (case A) and lunar gravities (case B) \((g = 9.81 m/s^2 \text{ and } g = 1.62 m/s^2)\) [23]. One of their qualitative observation was that the sinkage is higher on the Moon than on Earth (cf. Fig. 2–17). One could argue that on the Moon, the load applied on the wheel should be divided by six as well, to account for the lower weight on the Moon of a rover with a constant mass. This could lead to a lower sinkage discrepancy. Similar simulations were run, involving a 100mm-diameter wheel with 36 4mm-height grousers, using the torque-controlled approach. The simulation system was very similar to the one presented in the previous section, the soil bin sides being set as periodic boundaries. The plug-in PolyTorque was used, applying a second-order polynomial torque on a 4mm-thick particle wheel. Three cases were considered:

- Case A: a 0.3N vertical load was applied to the wheel, subject to terrestrial gravity.
- Case B: a 0.3N vertical load was applied to the wheel, subject to lunar gravity.
- Case C: a 0.05N vertical load was applied to the wheel, subject to lunar gravity.

In keeping with Nakashima et al., the sinkage was indeed higher on the Moon (case B), but is was still higher when the applied load was divided by six (case C), illustrating the lower weight of the rover in a lower gravity, as shown in Fig. 4–12. The screenshots were taken at the exact same timestep, giving sinkage values of 2.10mm, 10.37mm, and 5.64mm for cases A, B and C respectively. This means that the sinkage of a given rover of constant mass would be higher on the Moon than on Earth, on similar soils.

However, the time needed to reach a given distance in the soil bin were 0.336s, 0.372s and 0.291s for the cases A, B and C respectively. The rover would travel faster on the Moon (C) than on Earth (A), if driven by the same torque. It is also interesting to note
the presence of soil projections in case C, which are not present under terrestrial gravity, highlighting the need for dust protection on the rover.

4.2.3 **Comparison to non-DEM based wheel-soil interaction simulations**

The European Space Agency (ESA) is planning to launch a 6-wheel rover to Mars in 2018, in a mission called ExoMars. The development of the rover has already started. The German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt, DLR) and EADS Astrium are working on the wheel design. Two papers published in the year 2010 by Schaefer et al. (DLR) and Patel et al. (EADS) disclose the numerical methods used to evaluate the performances of both flexible and rigid wheel concepts [28, 24]. In this section, some of their simulations based on FEM-inspired soil contact models are reproduced in the DEM environment to check the consistence of the approach.

Eight rigid wheel and eight flexible wheel configurations are considered by Patel et al. as first iteration design solutions. The solutions were evaluated on a reference soil with a displacement-controlled approach, with a fixed slip ratio of 10% and a constant load \( W = 240 \, N \), and four performance metrics were estimated and compared: reaction torque, net traction, motion resistance and sinkage, as shown in Fig. 4–13. Only two rigid wheel configurations (EXMR20-3 and EXMR25-3) were considered viable and are listed in this figure, among the eight initially tested.
Figure 4–13: Simulated performance comparison of 2 rigid and 8 flexible wheels using a non-DEM based interaction model [24]: torque $T$, net traction $DP$, rolling resistance $R_g$ and sinkage $z_0$. The slip is fixed to 0.1 and the wheels are subject to a vertical load of 240N.

As the DEM simulations here only deals with rigid geometries, the rigid wheel EXMR20-3 was the one selected for replication and testing. In Patel et al., this wheel is described as a 20cm-diameter and 10cm-wide groused wheel, with 32 simple rectangular 1cm-height grousers. Its motion is fixed, with a slip ratio of 0.1, and the wheel-soil interactions are recorded. Thus, the displacement controlled approached described in Sec. 4.1.1 was used to model it. To reduce the number of particles involved in the simulation, a wheel ten times thinner was modelled. The system consisted of a 1cm-wide wheel, rolling on a virtual soil made of 29,752 particles in a 2cm-wide and 60cm-long soil bin. The servo-control feature, cited in Sec. 4.1.1, was used to add a 24N vertical load on the wheel. The motion of the wheel was set at a $v = 5cm/s$ horizontal velocity, combined with a rotation at $\omega = 0.556rad/s$, to insure a 10% slip (cf. Eq. 2.6). The soil used in the tests detailed by Patel et al. (“MPF mixed drift-cloddy” soil) had a cohesion of 220Pa and an internal

85
angle of friction of $33.1^\circ$. The DEM soil parameters used in the simulation to model this soil were obtained using the calibration algorithm, with an angle of repose target value fixed arbitrarily at $33.1^\circ$, as it was not mentionned by Patel et al. The simulation time was 5s, requiring a runtime of approximately 8 hours. A side view of the simulation system is shown in Fig. 4–14.

![Side view of the simulation system](image)

Figure 4–14: Side view of the DEM analog wheel-soil simulation of the EXMR20-3 rigid wheel cited in [24]. The wheel width as been reduced to 1cm in order to decrease the number of particles required for the soil.

The simulation predicted an estimated net traction of $5.96N$. Given the fact that the wheel was ten times thinner than the original one, an extrapolation of this value leads to a $59.6N$ traction for the 10cm-wide wheel, if the side effects are assumed as negligible. This result is in good agreement with the $\approx 70N$ value estimated by the mean of non-DEM based simulations in Patel et al. (cf. Fig. 4–13, wheel EXMR20-3). The estimated sinkage value was not as close though: the simulation predicted a sinkage around $17mm$, whereas it almost reached $50mm$ (one fourth of the wheel diameter) in Patel et al. This could be due to the low compressibility of the virtual soil, and could be fixed by adding compressibility to the list of properties in the calibration methodology (cf. Sec. 5.2). The torque could
not be compared as the torque computation in EDEM was found to have an error in a subroutine.\(^1\)

In Schäfer et al., more results about rigid wheel performance estimates are given and compared to experimental data [28]. A plot of the net traction, mentioned as drawbar pull in Schäfer et al., as a function of the slip is given, as well as the net traction vs. time, for a given slip (cf. Fig. 4–15). In this figure, the experimental net traction is almost independent of the slip, growing from 230N for a 10% slip, to 290N for a 70% slip. Conversely, their simulated results depend highly on the wheel slip. The net traction grows from -20N for \(s = 10\%\) to 300N for \(s = 50\%\), and remains almost constant at higher slip. This asymptotical trend is predicted in Bekker and Wong’s theories [3, 32], and is confirmed experimentally and numerically by Nakashima et al., as shown in Fig. 2–11 [23], which contradict the experimental results at low slip presented in Fig. 4–15 into question.

The displacement-controlled approach for DEM wheel-soil simulations allows us to reproduce these net traction measures for different slip values. As the applied load and the wheel geometry are not mentioned by Schäfer et al., it was decided to re-use the simulation deck reproducing the wheel tests presented in Patel et al., changing only the angular velocity to reach various slip values. The goal was to get the general traction vs. slip trend, without trying to validate quantitatively the results shown in Fig. 4–15. The translational velocity of the wheel was set to \(v = 5cm/s\) and nine different slip values were considered, from 10% to 90%. For each slip value, three 5 second simulations were run and the average net traction of these 3 runs was computed. Each simulation took roughly 8

\(^1\) EDEM does not update the position of the center of mass of moving geometries in the torque computation. This bug has been reported to EDEM Support and will be fixed in the next version.
Figure 4–15: Experimental and simulated average net traction of the ExoMars rover chassis, pulled at a constant speed of 1.08 cm/s, as a function of slip [28]. A full net traction measure as a function of time is given in the subdiagram, showing the great fluctuations of the net traction mainly due to the presence of grousers.

hours to compute, which gives a total simulation runtime of approximately 10 days. The net traction vs. slip curve obtained is given in Fig. 4–16.

Although the trend obtained for slippage under 30% is in good agreement with the expected asymptotical trend, the net traction falls at higher slip. This unexpected drop stems from the high sinkage of the wheel at high slip. Indeed, the simulation system involved a thin layer of soil (cf. Fig. 4–14) to reduce the number of particles and thus the simulation runtime. At high slip, the wheel literally dug into the soil and reached the bottom of the soil bin, as shown in Fig. 4–17. As a consequence, the reaction force on the wheel mainly came from the rigid bin and the net traction, computed as the horizontal component of the reaction force of the soil particles on the wheel, dropped.

The erroneous simulations at high slip were not re-calculated with a thicker soil layer for two reasons. Firstly, it would have required to double or triple the number of particles involved, multiplying the runtime by the same factor, which was not compatible with the project time constraints. Secondly, although such slip values are frequently reached by
Figure 4–16: Simulated net traction of a 32-grouser wheel subject to a 24N load, rolling at 5cm/s, as a function of slip.

off-road vehicles such as rally sport cars or buggies, they are completely out of the domain of planetary exploration rovers for control and safety reasons. The usual slip values in rover applications are kept as low as possible, and certainly below 30%.

The results at low slip ($s \leq 30\%$) were confirmed by the good correlation with the results of Patel et al. at 10% slip, as previously mentioned in this section, and by raising trend obtained. The predicted traction of -20N predicted by Schäfer et al. at slip 0.1 in Fig. 4–15 is, on the contrary, negative and far from the +230N traction experimentally measured, which raises some questions on the validity of their simulation model at low slip.

As a general remark pertaining to these simulations, one should keep in mind that the dynamic behavior of a loaded wheel is subject to high amplitude fluctuations, and performances such as net traction or sinkage must be computed as averages on runs of several seconds at the least, as confirmed the Schäfer et al. traction simulation in Fig. 4–15 (subdiagram) and the presented DEM simulations (cf. Fig. 4–18). The presence of grousers is mainly responsible for these fluctuations, but EDEM’s servo-control feature
used in the displacement-controlled simulations induces a high instability as well. The traction pulsing is observed at the same frequency as the grousers come into contact with the ground \( \left( f = N_g \times \omega/(2\pi) = 3.64\, \text{Hz} \right) \) in Fig. 4–18.

### 4.2.4 Comparison to Bekker equations

In Sec. 2.1.2, Bekker and Wong models for some of the wheel-soil interaction characteristics, such as gross traction, wheel soil resistance or sinkage, were described as a function of wheel parameters, soil properties and motion characteristics. Radziszewski et al. expressed these theoretical performances as a function of wheel parameters in the case of the lunar soil \([26]\). These expressions are used here to compute the theoretical slip and sinkage of the reduced scale vehicle presented in Sec. 4.2.1 on a lunar soil, and are compared to simulated values. Since slip and sinkage depend mainly on the wheel diameter and on the load \( W \), three simulations with various loads were run.

In Fig. 4–19, the estimated and simulated slips are compared, while in Fig. 4–20, the sinkages are compared. The first observation is that the order of magnitude and the trends match in both figures. The simulated sinkage is higher than expected, but the discrepancy

---

\( 2 \) EDEM Support does not support nor recommend the use of this feature anymore.
Figure 4–18: Simulated net traction of a displacement-controlled 32-grouser wheel subject to a 24N load \((v = 5cm/s, \omega = 0.7143rad/s, 30\% \text{ slip})\). The frequency of the main fluctuations corresponds to the grousers contacting the soil (here, \(f = N_g \times \omega/(2\pi) = 3.64Hz\), which leads to traction pulses every 0.275s).

is almost constant. As for the simulated slip, it is almost proportional to the weight, which is not the case in the predictions. The discrepancies come from several factors. First, the Bekker equations do not take grousers into account. They model grousers by an increase of the wheel diameter, as explained in Sec. 2.1.2. This could explain the under-estimation of the sinkage, as a groused wheel has a smaller contact area with the ground. Second, the soil parameters used by Radziszewski et al. and characterizing the lunar regolith are different than the ones used here to calibrate the soil. For instance, the Bekker equation linking sinkage and rover weight includes the soil exponent \(n\) (cf. Eq. [2.10]), which is unknown for the virtual soil.

To conclude, the Bekker semi-empirical equations can give a good approximation of the wheel-soil interaction characteristics for rigid circular wheels, provided that all soil
Figure 4–19: Comparison between the slip estimated with Bekker’s equations and the simulated one, for various loads, on a regolith soil.

parameters are known. Extrapolating these results to more complex wheel shapes rolling on partially characterized soils is not possible, though the trends and the order of magnitude of the results remain valid.

4.3 Tread shape optimization

The validation of the wheel-soil interaction model in the previous section gives an alternative to costly wheel prototype testings. Moreover, the time needed for each iteration in the design process can now be virtually reduced to the wheel-soil simulation runtime, skipping the manufacturing time. This could lead to an automated optimization of the tread shape provided two conditions are met. Firstly, wheel performances must be identified and measurable by the simulation output to be able to rank the different design solutions, as addressed in Sec. 4.3.1. Secondly, the DEM software should have an interface protocol allowing a third-part program to run a given simulation input deck and to obtain the output data, as detailed in Sec. 4.3.2.
4.3.1 Existence of optimum grouser configurations

The investigation for an optimum wheel is implicitly related to one or more objectives, involving measurable performances. In the lunar rover wheel case, the objectives depend on the mission scenario. The payload and the reliability requirements for an unmanned rover carrying a few scientific instruments or a pressurized rover capable of carrying 4 astronauts and towing a trailer are completely different, leading to different loading or shock absorption objectives. The scenarios are not yet known, limiting the objectives of this study to a proof of concept. To demonstrate that the simulation environment can prove the existence of an optimum tread shape for a given objective, arbitrary simulation sets were constructed, and the performances of various grouser shapes were compared pertaining selected criteria.

First, the simulation set cited in Sec. 4.2.2 and involving a torque-controlled 10cm-diameter particle wheel. A 20cm-long and 4mm-wide sandbox was used. At the beginning
of the simulation, the wheel is placed just above the soil, at the left end of the box, with no initial velocity. An external load is set at $t = 0\, s$, and a polynomial torque is applied once the wheel reached its static sinkage position, with the plug-in PolyTorque (cf. Appendix J). The simulation ends when the wheel reaches the right end of the box. Informations such as position, time and energy needed to reach the end of the box, translational or rotational speed at the end, are extracted then. In Fig. 4–21 for instance, the horizontal position of wheels having 0, 18 or 36 4mm grousers is plotted as a function of time (the position of the wheel at $t = 0\, s$ is set to 0). This figure shows that the 18-grouser wheel is

![Figure 4–21: Horizontal position of wheels with various number of grousers $N_g$, subject to identical polynomial torques, as a function of time. The grousers are 4mm-height and the wheels are subject to a 1N vertical load.](image)

...the fastest to reach the end of the box. Thus, more is not necessarily better for the number of grousers $N_g$. A wheel needs to have enough grousers to maintain a high shear stress in the soil, but as the shear stress is limited to $\tau_{\text{max}}$ (cf. Eq. [2.1]), there is a threshold after which adding more grousers does not increase the traction performance. On the contrary,
adding weight and inertia increases the energy needed to bury the grousers in the soil as the wheel rolls, thus leading to a lower overall performance.

The effect of grouser height was investigated by using the same simulation set and a 36-grouser wheel. As the driving torque is known, the energy required by the wheel to complete its journey could be computed. The results, given in Fig. 4–22, show the presence of an optimum grouser height of 5mm, leading to a minimal energy consumption of 5.05 J.

![Figure 4–22: Energy consumption of a 36-grouser wheel to reach the end of the sand box, as a function of the grouser height.](image)

The estimated energy given in Fig. 4–22 is the energy needed to put the wheel (and the rover) in motion from a static position. In the case of a small rover, required to stop and restart several tens of times a day, as it takes pictures or performs soil analysis for example, this value should be as low as possible. However, other objectives should be considered. Longer grousers mean a higher stress concentration at the junction with the
wheel hub, raising reliability issues; higher diameter and mass increase the launching space and payload requirements.

The wheels of several rovers, such as Spirit and Opportunity (cf. Fig. 2–5), were similar to the ones modelled here, with simple rectangular grousers. They can be almost completely defined by only three parameters - radius $R$, number of grousers $N_g$ and grouser height $h$ - and are easy to manufacture. Thus, the tests to find the optimum design should have been easy to plan. However, the simulation environment presented here would have saved a few prototype iterations.

To explore the 3D capabilities of the software and investigate complex tread shapes, the simulation set presented in Sec. 4.2.3 to reproduce the tractive performance analysis of rigid wheels for the ExoMars rover was used, this time with a 135mm-diameter displacement-controlled wheel [24, 28]. A first series of 19 grouser configurations were compared, varying the number $N_g$, the height $h$ and the angle $\alpha$ of the grousers. It was found that $(N_g = 18; \alpha = 0^\circ; h = 8\ mm)$ was the configuration giving the highest traction at 5% slip, as shown in Fig. 4–23.

These results illustrate the increase in performances that one can expect by optimizing a parametrized wheel. However, the design of experiment can be improved. As each simulation takes hours to complete; a full factorial design of experiment is not possible, especially within the 8-dimensional design space of the parametrized geometry wheel considered. An external program can implement an online approach of the design space exploration, by computing promising sets of design variables from a database, launching the simulation, and getting the results to update the database (cf. Sec. 3.4.1), as explained in the next section.
4.3.2 Building an optimization algorithm for optimum 3D grouser design

In this section, the general architecture of a MATLAB code for automated wheel tread optimization process is presented, using displacement-controlled simulations and based on the parameterization described in Sec. 4.1.1. For the sake of simplicity, a set of parameters will be generically called a point of the design space, and the process of building the simulation input deck with a wheel parametrized by a given point, running the simulation, and getting the extracted data file will be called a point evaluation.

The optimization process is based on the RSM-workflow presented in Fig. 3–6. Fig. 4–24 updates this workflow for this specific wheel case. The workflow starts with step (0), consisting of the design space definition, the determination of an initial sample of points, and the evaluation of these points to build the initial database. The definition of the design space, i.e. the boundaries between which each design variable should remain, is directly related to the rover mission scenarios. The wheel radius and width ensue from the payload

Figure 4–23: Traction of 135mm-diameter displacement-controlled wheels at 5% slip for various grouser configurations.
supposed to be carried by the wheel. From this first configuration, the number of grousers, grouser height, grouser thickness, can be bounded and a list of geometrical constraints can be set, forbidding the number of grousers times the grouser width to be larger than the wheel circumference for instance. Then, an initial sample of points must be determined. Several design of experiment techniques are briefly described in 3.4.1. In a turbine blade design case study, I. Lepot suggests for instance to use an initial population of 3 to 5 times the number of design variables [17]. Each one of these points is then evaluated and its
performances are stored in a database (these two stages are detailed in steps 2 to 5). The simplified MATLAB code for this step is given below:

```matlab
function [] = AutomatedOpti()

% STEP 0.
% Definition of the design space: the minimum and maximum values for each parameter
% are defined to set the design space boundaries:

point_min_values = [Ng_min R_min h_min ...];
point_max_values = [Ng_max R_max h_max ...];

number_of_parameters = length(point_min_values);
boundaries = [point_min_values; point_max_values];

% The function Sample() is called to determine an initial sample of points to be
% evaluated. This sample depends on the design space boundaries:

initial_sample = Sample(boundaries);

% Each one of these points is evaluated through the function Evaluate() and its
% linked performance stored in the database (the function Evaluate() is detailed
% in steps 2 to 5).

for i = 1:length(initial_sample)

    % Evaluation of the point:
    performance_i = Evaluate(initial_sample(i));

    % Update of the database. Here the database is a simple TXT file. Each line
    % of the file contains the point parameters followed by the associated
    % performance.
    update = [initial_sample(i) performance_i];
    dlmwrite('database.txt', update, '-append','delimiter','

end

...
% STEP 1.
% This step is the first step of the optimization loop, which includes steps 1 to 6.
As a consequence, all these steps are included inside an infinite loop, which stops only when the objectives are reached:

while(true)
    % Read the database:
    database = dlmread('database.txt');
    % Get the next point to evaluate with the function GetNextPoint() (this function, not detailed here, builds the surrogate model of the performance response surface from the already evaluated points of the database and returns the point optimizing this response inside the design space boundaries):
    next_point = GetNextPoint(database, boundaries);

Steps (2) to (5) consist in the point evaluation, grouped in the function Evaluate(). First, it assembles the XML file defining the simulation input deck with the wheel geometry built according to the promising parametrization computed previously in step (1). The function CAD_to_XML(), cited in Sec. 4.1.1 and given in Appendix H can be used. It has 12 input parameters, but a simpler or refined parametrization could be used for more complex wheel geometry. Then, the XML simulation deck prepared is computed through EDEM via a batch file in step (3), and the file containing the extracted data is read in step (4). The performance of the wheel is computed from these extracted data next, in step (5), and returned to the main function. In Chap. 4, the performance criteria were the mean traction or the power consumption for instance. The Evaluate() function is called by the root function and given in a different frame for clarity purpose:

function [point_performance] = Evaluate(point)
The input argument of the function `Evaluate()` is the point to be evaluated, the output is the point performance. This function encompasses steps 2 to 5:

**STEP 2.**
Construction of the XML file describing the simulation input deck with the chosen wheel geometry:

```matlab
CAD_to_XML(next_point);
```

**STEP 3.**
The batch file is launched:

```bash
! batch.bat
```

This batch file can be as short as the single-line command: `edem -console -i "simulation.XML" -p 8 -r 5 -w 0.01 -g 5 -t 8e-06 -e "dataExtraction_dem.cfg"`

It essentially tells EDEM to run the simulation and to extract the data according to the extraction configuration `dataExtraction_dem.cfg` when the simulation is over.

**STEP 4.**
The file containing the data from the simulation is read. EDEM saves the data in CSV files:

```matlab
point_data = dlmread('dataExtraction.csv');
```

**STEP 5.**
The performance is computed from the extracted data (different performance criteria, not detailed here, can be used, such as mean traction or power consumption):

```matlab
point_performance = GetPerformance(point_data);
```

Then, the main function checks the performance of the evaluated point, and either stops the loop if this performance meets the requirements, or restarts the loop after having updated the database (step 6):

```matlab
% STEP 5 (continued).
% The new point and its associated performance are added to the database:
update = [next_point, performance];
dlmwrite('database.txt', update, '-append', 'delimiter', ',');

% If the objectives are reached, the loop is stopped and the optimum point of the database is returned.
if (ObjectiveReached() = true)
    % Find the maximum (or minimum) value in the database points' performances (last column of the database table):
    [max_perf, indice] = max(database(:, number_of_parameters + 1));
    % Print the database line with optimum performance:
    optimum_line = database(indice, :);
    optimum_point = optimum_line(1:number_of_parameters);
end
```
The process described here was valid for a single objective optimization, in which the performance of a given point in the design space is fully described by a scalar. This allows the computation of a surrogate model of the response surface. But if several conflicting objectives should be optimized, a multi-objective optimization algorithm is to be used, as explained in Sec. 3.5.1. In Sec. 3.5.2, an a-priori MOO method was used by minimizing the weighted least-square error between the three soil properties actual values and their target values, thus reducing the problem to a single-objective optimization case, but as cited previously, a more computationally expensive a-posteriori method computing the Pareto front can be used as well. The optimal solution is then chosen by the user among this Pareto front.

The MATLAB architecture presented here was not implemented for several reasons. First, the rover requirements and its performance objectives were not available by the time of this research, thus it was not possible to set the design space boundaries and optimization objectives. Second, the construction of the MOO algorithm was not in the present study objectives and is a complex task in itself. Third, the current displacement-controlled approach requires almost one day of computational runtime per wheel design evaluation as EDEM’s servo-control instabilities require the computation of average performances on long simulations as explained in Sec. 4.2.3. Consequently, this algorithm is not included in the present study. However, a much more precise motion control solution given as a
recommendation in the future works section might decrease this evaluation time and allow automated tread shape optimization on a shorter time scale (cf. Sec. 5.2.2).

4.4 Summary

The constraints and possibilities of the DEM software in terms of geometry motion control leads to the consideration of two different approaches for the wheel-soil simulations. In the first approach, the wheel is considered as a geometry object, built as an assembly of triangular surfaces. Its motion is completely defined a-priori by the user, as well as the load, and the reaction efforts of the soil are extracted from the simulation. The second approach features particle wheels, made of an assembly of spheres. Although it prevents a precise wheel shape modelling due to its particle nature, the major advantage of this approach is that external inputs, reproducing the driving torque from a motor for instance, can be applied to the wheel, and its motion derives completely from the interactions with the soil. This allows the exportation of data such as sinkage, slip, velocity and energy consumption. For both approaches, a MATLAB function was written to automatically build the XML simulation input decks from a set of wheel and motion parameters, which allows an easy comparison of the performances of various wheels.

In the second part of the chapter, the validity of such simulations is demonstrated by comparing them to experimental and theoretical measures, and to other simulations based on non-DEM contact models. Although these comparisons brought to light significant discrepancies partially due to missing information on soil and wheel characteristics, the simulation measures are accurate in predicting performances, such as power consumption or net traction, when using the soil calibration tool described in Chap. 3.

This validation of the first 3D DEM wheel-soil simulations paves the way for DEM-based 3D design optimization. The last section of the chapter shows that optimum values can be found for wheel tread parameters, such as the number of grousers or their height,
by comparing performances of different wheels. Moreover, the structure of an automated optimization algorithm is proposed.
CHAPTER 5
Conclusions and recommendations for future works

5.1 Conclusions

Due to its relative newness, the discrete element method and any derived software package suffer from some limitations for the modelling of three-dimensional interactions between wheels and granular soils. The present thesis addressed these issues, proposing a numerical modelling methodology with the software package EDEM.

First, a literature search reviewed the progress over the last century in terramechanics and wheel-soil interaction theoretical modelling. The attempts to model them through two-dimensional DEM simulations in recent years led to good traction or sinkage predictions, although the lack of DEM soil calibration was highlighted.

Next, a four-step systematic soil calibration methodology, based on geotechnical measurements on actual soil, and repeated numerically, was proposed in Chap. 3 to minimize the discrepancies between the target soil properties and the virtual soil properties. With this methodology, the silica sand used in the test bed apparatus could be successfully modelled, with realistic angle of repose, cohesion and internal angle of friction. The fidelity of the lunar regolith modelling was not satisfactory due to its very specific properties, which could not be reproduced by the selected spherical soil particle.

Next, two different approaches for wheel motion control were proposed in Chap. 4, limited, to some extent, by the software. In the first approach, the wheel was considered as a geometry, built as an assembly of triangular surfaces. A MATLAB function, assembling the wheel as a function of eight tread shape parameters, was coded. The wheel motion was
completely defined a-priori by the user, and the reaction efforts of the soil were evaluated. The second approach featured particle wheels, made of assemblies of spheres. Although it prevented a precise wheel shape modelling due to its particle nature, this approach presented a major advantage, allowing the application of external forces on the wheel, using customized plug-ins. The wheel motion was thus computed as a function of the soil interactions. Several plug-ins were coded, to apply an external torque to reach a constant speed for instance. The validity of those two approaches was then demonstrated by comparison to other works. The power consumption of a rover rolling at a constant speed could be estimated with an error of less than 8% from the experimental measure at 52N load using the torque-controlled approach. The net traction of a 32-grouser wheel at 10% slip evaluated with the displacement-controlled approach was in good agreement with Patel et al. value, differing by 16%. The traction and sinkage variations highlighted by Nakashima et al. for a wheel under terrestrial or lunar gravities were confirmed.

Finally, the three-dimensional DEM simulation environment developed in this study could prove the existence of optimum values for some tread parameters such as the number of grousers or their height by comparing performances of different wheels. The energy consumption of a 10cm-diameter and 36-grouser wheel subject to a 1N load was found to be minimum with a grouser height of 5mm. On a 135mm-diameter wheel, the traction performance at 5% slip of 19 grouser configurations was evaluated and an optimum configuration was identified. Moreover, the structure of an automated optimization algorithm was proposed.

The objectives of this study, detailed in Sec. 1.2 as (1) defining and developing a numerical modelling and analysis methodology of the wheel design, based on the discrete element method, to be used for tread shape optimization, and (2) validating this methodology by comparing simulations results to experimental ones, were addressed successfully.
However, this simulation environment cannot yet be used in the global project of compliant wheel design due to EDEM incompatibility with compliant geometries. It is the hope of the author that the possible improvements listed as recommendations in the next section will soon be available, allowing accurate dynamics control and compliant design compatibility.

5.2 Recommendations for future works

This section highlights and suggests some perspectives for future works in wheel-soil DEM simulations, with the current EDEM version or with more advanced DEM softwares. First, some ideas are proposed to enhance the soil modelling. Then, a coupling module allowing a new motion control of the geometries is presented. Finally, the possibilities that could stem from the development of a coupling between a discrete element software and a finite element solver for compliant wheel design are described.

5.2.1 Improved soil modelling

The soil modelling method presented in Chap. 3 and based on a methodology calibrating three soil design variables ($\mu_s, \mu_r, r$) to target three soil properties values was used successfully to model the silica sand used in the test bed, as detailed in Sec. 3.5.4, and the Mars simulant used by Patel et al. for the ExoMars wheel tests (cf. Sec. 4.2.3). Moreover, it is very simple as it involves one spherical particle prototype only. However, it was not able to accurately model the lunar regolith’s special properties. Indeed, Tab. 3–6 shows that the high lunar regolith angle of repose could not be reproduced by the virtual soil. It was out of the response surface of the virtual soil, meaning that no feasible set of design variables could satisfactorily bring the virtual soil behavior close to the real one.

Although simulations involving virtual silica sand were validated experimentally, confirming the soundness of the calibration methodology, this fact questions the validity of the simulations involving regolith. For this reason, a more complex soil modelling should be developed.
A first step to improve the virtual soil fidelity could be the increasing of the number of target properties. In this study, cohesion, internal angle of friction and angle of repose were the only properties considered. They were chosen because they play important roles in terramechanics and required only two experiments for their determination, these experiments being simple to simulate. The following additional properties might be used in the future:

- **Compressibility.** This property is important for lunar regolith, which is a fine and highly compressible dust. The Apollo 15 report mentions that the high compressibility induced a high sinkage of the Lunar Roving Vehicle, leading to a decreasing maneuverability [6]. Compressibility is also important for the accurate modelling of multi-passes simulations, in which the rear wheel rolls in the tracks of the front one, and is thus subject to different interactions. Moreover, the compression test is simple, and it should not be a problem to simulate since it involves only the application of a load on a soil sample in a cylinder.

- **Bulk density.** This property is closely linked to the bearable capacity of a soil, and thus to the wheel sinkage. Tables of bulk densities are available in the literature even for lunar regolith [11], and the measure of a virtual soil bulk density is possible in EDEM as one can select the particles in a given domain and get their total volume and mass.

- **Slope stability.** This property is different than the angle of repose as it gives the stability of consolidated soils for a given slope. The behavior of wheels on sloping terrain was not investigated in the present study but will need to be studied in the future to evaluate the critical slope that a given rover can overcome. Craters are often cited as possible landing sites on the Moon as they provide a natural protection
and they give access to deeper geological strata, which makes the trafficability on slopes a major concern for future lunar rovers.

- **Particle size distribution.** The regolith size distribution is well-known, and it will play a major role in reliability issues, especially for compliant wheels with several moving parts. Indeed, the abrasiveness and the microscopic size of such particles is responsible for a very fast wear and dangerous clogging, according to Apollo reports [6]. A simulation system consisting of an entire wheel rolling on a soil cannot involve such small particles, due to the limits in computational power. However, it is possible to simulate the interactions between regolith and just a joint between two mechanical pieces.

Although this list of target properties could be extended further, the fidelity of a soil calibrated to fit 4 or 5 of them would certainly give satisfactory terramechanics prediction. However, if the number of target properties is increased, a higher number of design variables is required to be able to be close to each of them. Soil properties, such as size distribution around the mean particle radius, Poisson’s ratio, shear modulus and density, were found to have a relatively small influence on the soil angle of repose (cf. Tab. 3–3), but they might have a greater influence on other properties (the compressibility can depend on the shear modulus and Poisson’s ratio for instance). Also, more complex multi-sphere particle prototypes could be used to illustrate the high roughness and complex shapes of soil such as lunar regolith. Li et al. used a combination of 4 different particle prototypes in their regolith two-dimensional DEM modelling for instance, as shown in Fig. 5–1 [31]. By using several particle prototypes, the proportion of each of them can be introduced as a new soil design variable. Lunar regolith could be modelled with a higher proportion of highly angular particles than silica sand for example. Compressibility and cohesion are very dependent of particle shapes, as it is linked to particle interlocking. A last solution
for fitting slope stability and bearing capacity values is the use of different particles in the subsoil. Bigger spheres under the superficial soil layer should increase the soil stability.

The virtual soil modelling is a crucial issue if a realistic terramechanics simulation is desired, but one important limitation in EDEM for this study was the control of the wheel, or in other words the geometry management by the software, and a possible improvement in dynamics control is addressed in the next section.

5.2.2 Enhanced wheel dynamics control

As explained in Chap. 4, the major drawback of EDEM in geometry control is that the dynamics of the geometries have to be set in the pre-processing step and do not depend on the interactions with the soil. This problem was bypassed by building particle wheels, but this solution is not completely satisfactory due to the poor design possibilities of such
wheel modelling. What is really needed to model accurate terramechanics simulations is an effort-controlled geometry dynamics capable of simultaneously applying external efforts, such as driving torque and loads, and taking into account the reaction forces of the soil.

ESTEQ Engineering, a South-African software developer specialized in computer-aided engineering, developed and released in 2009 the coupling module *EDEM Dynamics Coupling*, which computes the geometry dynamic response to soil interactions [9]. It includes a coupling server which, at every given number of DEM timesteps, extracts the forces on the geometry from EDEM and inputs them in the dynamics package, gets the geometry displacement, and finally updates the geometry position in EDEM to let it continue the DEM computation. According to DEM Solutions, an update every ten timesteps gives acceptable results and does not augment the computation runtime significantly. MSC ADAMS is the dynamics package recommended by ESTEQ. ADAMS is a multi-body motion analysis software which can be used to evaluate the power requirements or tractive performances as previously presented in the simulations, but it also has capabilities in fields such as vibration and durability analysis as well, which are important issues in the design of planetary rover, although not addressed in the present study.

This coupling module has one major drawback to be mentioned: the actual version does not support the XML input deck definition used in EDEM. This limitation excludes the use of automated optimization algorithms as the one described in Sec. 4.3.2. Such a routine could be implemented in the future versions.

### 5.2.3 Investigating compliant wheel design with FEA-DEM coupling

Another field of improvement to DEM simulations would be to compute the deformations of the geometries due to the action of the granular material on them. Indeed, this would allow the investigation of compliant wheels, which are particularly adapted to off-road locomotion. Fig. 4-13, comparing the performance of 2 rigid (on the left) and
8 flexible (on the right) wheel concepts for the ExoMars rover, clearly shows the better tractive performances of the flexible wheels, although flexibility brings other issues such as reliability and resistance to wear. The computation of the deformation of solids is a well mastered field in modern engineering, using the finite element analysis (FEA). Coupling DEM and FEA is a complex task. It implies to import the geometries and the boundary conditions applied to them (the forces of the soil on the geometries) from the DEM software to the FEA one, compute the geometry deformations using FEA, and re-input the updated geometries in the DEM simulation system. Ideally, this would be done at each timestep. This additional step could be included in the DEM loop just before the update of the geometries and particles positions, as shown in Fig. 5–2 as opposed to Fig. 2–6 without the FEA coupling.

Such a coupling would not only require an automated data transfer between the two software packages, but it would greatly increase the simulation runtime as well. As the geometry shapes would be different at each time step, the FEA solver would need to re-mesh
them every time. In 2009, Lee et al. proposed an initial draft of this promising application by automating the computation of the deformations of a chute used to convey rocks [16]. They wrote a routine extracting the forces of a constant rocks fall on the chute once the steady state is reached, and inputing them in ANSYS to compute the chute deformations (cf. Fig. 5–3). They call this coupling a “one-way quasi-steady-state” coupling, meaning that the information about soil-structure interactions goes only from the DEM software to the FEA one, once the steady-state is reached.

Figure 5–3: An example of DEM-FEA coupling proposed by LEAP Australia and based on the softwares EDEM (DEM part) and ANSYS (FEA part): the soil-structure interactions are computed with EDEM (left), then the efforts on the structures are extracted (middle) and finally the geometry deformations are computed through finite element analysis (right) [16].

Although the word “coupling” is misused here, as the information goes only one way, this routine can be useful in systems in which the geometry is subject to a constant interaction with a granular flow, and in which the deformations are in the low-strain and elastic domain only. For high-strain deformations, the structural change of the geometry will affect the granular flow, which will lead to different interactions, and thus different geometry deformations, requiring a full two-way coupling. In a wheel-soil simulation involving a compliant wheel, a two-way coupling is necessary to update the wheel deformations as it rolls on the soil. The area of application of the soil reaction forces on the wheel is constantly
moving. Consequently, the boundary conditions never reach a steady-state. Moreover, wheel concepts with high compliance are usually subject to high-strain deformations.

To conclude, the relatively newness of the DEM softwares and the lack of file standards in the DEM field limit the development of two-way FEA-DEM coupling presently, but as the field matures and with the ever increasing computational resources available, combined with a growing interest of researchers demonstrated by the work of Lee et al., a version of ready-to-use DEM-FEA solution might become available in the coming decade, with promising applications in compliant wheel design and optimization.
Appendix A: The different contact models in DEM

The first DEM contact model, developed by Cundall and Strack, is briefly described in Sec. 2.2.1. In this model, several input parameters come into play in the definition of contact forces: $K_n$, $K_t$, $C_n$, $C_t$, $\phi$ and $c$ (cf. Eqs. [2.20-2.24]). In fact, Cundall defined the damping coefficients $C_i$ as proportional to the stiffnesses $K_i$ with a proportionality constant $\beta$:

$$C_n = \beta K_n \quad , \quad C_t = \beta K_t.$$  \hspace{0.5cm} (A-1)

This reduces the number of input parameters to 6: $K_n$, $K_t$, $\beta$, $\phi$ and $c$, plus the material density $\rho$.

In 1992, Tsuji et al. used a more sophisticated model based on the non-linear Herz model for the normal force, and on a simplified version of Mindlin and Deresiewicz solution (assuming “no-slip” between particles) for the computation of the tangential force [30]. Better than varying linearly, as in Eq. [2.20], the elastic component of the normal force $\vec{F}_n^e$ varies with the 3/2 power of the normal overlap $\delta_n$ according to the Herzian theory:

$$\vec{F}_n^e = K_n \delta_n^{3/2}$$  \hspace{0.5cm} (A-2)

in which the stiffness unit changes. $F_n^e$ is written as a scalar for the sake of notation simplicity.

In addition, the elastic component of the tangential force $\vec{F}_t^e$ is taken as the no-slip case of Mindlin and Deresiewicz’s theory (MD) describing a new model for the tangential force of particles in contact, under oblique loads [21]:

$$\vec{F}_t^e = -K_t \delta_n^{1/2} \vec{\delta}_t$$ \hspace{0.5cm} (A-3)
Moreover, they use a different damping coefficient, assumed to be the same in both normal and tangential directions:

\[ C_i = \alpha \sqrt{mK_n \delta_{in}^{1/4}} \]

where \( \alpha \) is a function of the coefficient of restitution \( e \), measured experimentally as the ratio of the relative velocity \( v \) between two contacting particles just after impact with respect to the relative velocity just before impact \( v_0 \): \( e = v/v_0 \). As Tsuji et al. considered their material as cohesiveless, the Coulomb criterion used in Eq. [2.24] was simplified to: \( (F_t)_{max} = \mu F_n \) with \( \mu = \tan \varphi \). Consequently, their model requires 5 input material properties: shear modulus \( G \), Poisson ratio \( \nu \), coefficient of restitution \( e \), coefficient of friction \( \mu \) and density. This model will be referred to as \( H-MDns \), for Herz-Mindlin and Deresiewicz, with the “no-slip” assumption.

Tsuji’s \( H-MDns \) model is obviously more complex than the linear spring model (with longer simulation run time as a direct consequence), but it still simplifies the computation of \( \vec{F}_t^e \) by using the no-slip approximation in Mindlin and Deresiewicz’s model. The novelty of Mindlin and Deresiewicz’s theory lies in the loading history dependence of \( \vec{F}_t \), whereas in linear spring and H-MDns models, \( \vec{F}_t \) is derived immediately from the instantaneous overlap and relative velocity, without considering any history effect. To verify the validity of the “no-slip” approximation, Di Renzo implemented in 2004 the complete MD theory for \( \vec{F}_t^e \) in his own DEM code [27]. Although this third contact model, noted as \( H-MD \), will not be detailed here, Di Renzo’s work is worth citing because he compared the prediction of the 3 contact models - linear spring, H-MDns and H-MD - to nearly exact analytical solutions at a microscopic level, and to existing experimental data at a macroscopic level. He concluded that the linear spring model was satisfactory at a macroscopic scale, with no significant improvements when using H-MD or H-MDns models. However, the results at a microscopic scale showed a high dependence on \( K_n \) value when using the linear spring
model, and recommended the use of more sophisticated models such as H-MD and H-MDns. This issue raises the problem of material property calibration. Indeed, in addition to density $\rho$ and static friction coefficient $\mu = \tan \varphi$, the linear spring model needs four input parameters ($K_n$, $K_t$, $\beta$ and $c$) which do not all correspond to measurable properties of the material, and thus are difficult to tune. On the contrary, H-MD and H-MDns models need three input properties ($G$, $\nu$ and $e$) corresponding to three physical properties.

The calibration of the numerous contact model parameters is a key step for accurate DEM simulations. Appendix B gives an example of DEM calibration available in the literature.
Appendix B: An example of DEM parameter calibration

To model accurately the dynamic behavior of granular materials such as ore or seed grains, and their actions on conveyors, silo walls or machines by the mean of the discrete element method, one has to calibrate the parameters describing the interactions between particles at a microscopic level, i.e. coming into play in the contact model. To illustrate this issue, the calibration done by Coetzee et al. to model interactions between corn seeds and rigid structures follows [5].

In their paper published in 2009, Coetzee et al. simulated the behavior of corn seeds (the granular material) during a silo discharge and a bucket filling. They modeled the corn seeds with two-dimensional DEM based on Cundall’s work, using the linear spring contact model. The virtual material modelling requires the determination of a few design variables. First, the particle size, shape, and size distribution have to be defined. To model the corn seeds, they used real-size particles composed of two overlapping disks (cf. Fig B-4). Six material properties have to be input to the contact model, as detailed in Appendix A: the particle stiffnesses $K_n$ and $K_t$, the damping coefficient constant $\beta$, internal angle of friction...
\(\varphi\), cohesion \(c\) and density \(\rho\). Since the material is non-cohesive, the cohesion was set to \(c = 0\), and Coulomb’s criterion was replaced by \((F_t)_{max} = \mu F_n\), introducing the material property \(\mu\) instead of \(\varphi\). \(^1\) Moreover, normal and tangential stiffnesses were assumed to be equal, and the damping coefficients were set to \(C_n = C_t = 0.7\) in the calibration simulations. The density was set to \(778 kg/m^3\) after weighting a given volume of material. Finally, the stiffness \(K\) and the coefficient of friction \(\mu\) were the only two material properties identified by the authors as requiring a specific calibration.

To calibrate these two properties, the real internal angle of friction \(\varphi_0\) of corn and the confined Young’s modulus \(E'_0\) were measured using the direct shear test and the confined compression test respectively: \(\varphi_0 = 23^\circ\) and \(E'_0 = 1.6 MPa\). These experiments were repeated numerically through DEM simulations for different sets of material properties \((\mu, K)\) and the values for \(\varphi\) and \(E'\) plotted as a function of \((\mu, K)\) (cf. Fig. B-5). As shown in Fig. B-5, \(E'\) and \(\varphi\) were measured for any \((\mu, K)\) in the set \(\{0.1, 0.12, 0.15, 0.2, 0.3\} \times \{10, 50, 100, 300, 450, 500\}\) with \(K\) expressed in kN/m. Consequently, 30 simulations were run for each test. Luckily, the confined Young’s modulus of the virtual material depended only on the stiffness \(K\). They could choose the stiffness value to match the experimental confined Young’s modulus \(E'_0\) (cf. Fig. B-5-a). \(K = 450 kN/m\) was considered as optimum by the authors. Then, \(\mu = 0.12\) was chosen to match the virtual material’s internal angle of friction to the experimental one, given that the particle stiffness was set to \(K = 450 kN/m\) (cf. Fig. B-5-b). They could thus calibrate the value of the design variables not previously determined.

\(^1\) One should notice that \(\mu\) denotes here the friction coefficient at a microscopic level, between two contacting particles, and thus it is not equal to \(\tan \varphi\). Indeed, \(\varphi\) is the macroscopic internal angle of friction, which has been shown to be higher to \(\tan^{-1} \mu\) due to the interlocking friction phenomenon [5].
This calibration was then validated by comparing experimental and simulated granular flows in three different experimental setups. The first one involved the measure of the angle of repose of a thousand grains dropped on a flat surface, for which the experimental value \((25 \pm 2^\circ)\) and the DEM one \((24 \pm 1^\circ)\) were in agreement. In the second setup depicting a dynamic silo discharge, DEM and experimental granular flow patterns were in good agreement, although the silo discharge was faster in the simulation (cf. Fig. B-6). Finally, filling percentages of a bucket digging into corn seeds could be accurately simulated as well.

To sum up, we can divide the calibration process presented by Coetzee et al. in four steps:

- Determination of the design variable characterizing the virtual material.
- Identification of those which need to be calibrated: \(\mu\) and \(K\).
Figure B-6: Comparison of experimental and simulated flow patterns occurring during a silo discharge [5].

- Experimental determination of two material properties, the confined Young’s modulus $E_0^\prime$ and the internal angle of friction $\varphi_0$, and simulation of these experiments for various sets of design variables ($\mu, K$).
- Optimization of $\mu$ and $K$ to minimize the discrepancy between the measured ($E_0^\prime, \varphi_0$) and the simulated properties ($E^\prime, \varphi$).

A unique couple ($\mu = 0.12, K = 450\text{kN/m}$) minimizing both differences $|E_0^\prime - E^\prime|$ and $|\varphi_0 - \varphi|$ was found, since these two objectives were not conflicting. Indeed, $E^\prime$ was not a function of $\mu$. In the calibration case described in Chap. 3, the objectives will be conflicting, a multiple-objective optimization approach will be used to find the design variable values minimizing the overall error on the discrepancies.
Appendix C: About the software EDEM

This appendix describes the software EDEM 2.2, used for all DEM simulations presented in this study. The software is divided in three modules: Creator (pre-processing steps), Simulator (computation of the simulation), and Analyst (post-processing steps).

In Creator, the user defines the simulation environment, composed of geometries and particles. The geometries, representing the rigid structures that will interact with the granular material, can be imported from CAD models or defined manually as combinations of simple shapes such as boxes or cylinders. The granular material is modelled by particles according to pre-defined particle prototypes. A particle prototype consists of one or more spheres, each one having its own radius and center position. Consequently, non-spherical real particles can be modeled by filling the particle shell with several spheres of different radii (cf. Fig. C-7). To create particles in the system, the user defines creation rules called factories, consisting in a volume in which the particle will be created, a generation rate (for continuous generation) or a number of particles (for instantaneous factories), a particle prototype, and other parameters such as size distribution, starting time, duration, initial particle position, orientation or velocity.

Each particle prototype or other geometry element is associated to a material $M_i$, defined by three properties. Moreover, three interaction parameters have to be defined for each possible couple of material $(M_i, M_j)$. All these input parameters are listed in Tab. 2–2, and Fig. C-8 shows an EDEM screenshot of the parameter input tab.

Once all elements and parameters of the system have been specified, one can run the Simulator. The contact model used to compute the interaction forces between two contacting spheres belonging to two different particles is based on H-MDns model described in Sec. 2.2.1. The detailed equations are given here, as a function of the material and
interaction parameters listed in Tab. 2–2 [29]. The normal components of the contact forces on a spherical particle \( i \) by a contacting spherical particle \( j \) are given by

\[
\vec{F}_n^e = -\frac{4}{3} Y^* \sqrt{R^* \delta_n} \vec{n}, \quad \vec{F}_n^d = -2 \sqrt{\frac{5}{6} \beta \sqrt{S_n m^*}} \vec{v}_{rel}^n
\]  

(C-5)

in which \( \vec{n} \) is the unit vector from the center of particle \( i \) to the center of particle \( j \), \( Y^* = \left( \frac{1-\nu_i^2}{E_i} + \frac{1-\nu_j^2}{E_j} \right)^{-1}, R^* = \frac{R_i R_j}{R_i + R_j} \) and \( m^* = \frac{m_i m_j}{m_i + m_j} \) are respectively the equivalent Young Modulus, radius and mass, \( \delta_n \) is the normal overlap, \( \vec{v}_{rel}^n \) is the normal component of the velocity of \( i \) relative to \( j \), and \( S_n = 2Y^* \sqrt{R^* \delta_n} \) is the normal stiffness. The normal damping force is proportional to the parameter \( \beta \), a function of the coefficient of restitution according to the formula

\[
\beta = \frac{\ln e}{\sqrt{\ln^2 e + \pi^2}}
\]  

(C-6)

The tangential components of the contact forces acting on \( i \) are given by

\[
\vec{F}_t^e = -S_t \delta_t, \quad \vec{F}_t^d = -2\sqrt{\frac{5}{6} \beta S_t m^*} \vec{v}_{rel}^t
\]  

(C-7)
Figure C-8: Definition of material and interaction parameters in EDEM. Regolith material properties and regolith-aluminum interaction coefficients are shown here.

in which \( S_t = 8G^* \sqrt{R^*} \delta_n \) is the normal stiffness, \( \vec{v}_{rel} \) is the tangential component of the velocity of \( i \) relative to \( j \), and \( \delta_t \) is the tangential overlap, or in other words the integral of \( \vec{v}_{rel} \) over time of contact. However, the Coulomb law of friction imposes an upper limit to the magnitude of the total tangential force. Indeed, if

\[
\left\| \vec{F}_t + \vec{F}_{td} \right\| > \mu_s \left\| \vec{F}_n + \vec{F}_{nd} \right\|
\]

(C-8)
then the total tangential force is no longer equal to the sum of the two terms of Eq. C-7, but is given by

$$\vec{F}_t = -\mu_s \left| \vec{F}_e + \vec{F}_d \right| \vec{v}_{rel}$$

(C-9)

Furthermore, EDEM has an artificial equation to its contact model, which adds the friction induced by surface roughness between contacting particles. Indeed, as the virtual particles are composed of spheres, they can potentially roll over each other frictionlessly. In reality, such motion implies energy losses due to the particle surface roughness and asperities. A torque, proportional to the rolling friction coefficient $\mu_r$, is thus induced by particle $j$ on the center of mass of $i$ in the direction opposed to the rolling motion between $i$ and $j$ to model the loss of energy due to friction occurring between real particles. The torque per unit of time is given by

$$\tau_i = -\mu_r \vec{F}_e D_i \vec{\omega}$$

(C-10)

in which $D_i$ is the distance of the contact point from the center of mass of $i$ and $\vec{\omega}$ is the unit angular velocity vector of $i$ relatively to $j$.

In addition to these forces, EDEM can manage electrostatic interactions or customized forces. The Application Programming Interface (API) allows the user to define its own forces to model more complex phenomena, such as stickiness or van der Waals forces. This feature will be used to apply an external torque to a “wheel” particle in Sec. 4.1.2.

The choice of the time step $\delta t$ in Simulator is of great importance, as mentioned in Sec. 2.2. $\delta t$ must be shortened until the convergence of the result is reached, as the mesh of an FEM simulation must be refined until convergence of the stress field for example. On the other hand, $\delta t$ cannot be too short as the simulation runtime is inversely proportional to it. EDEM recommends to set $\delta t$ as a percentage of Rayleigh time step $T_R$, which is the
time needed for a shear wave to propagate through a particle [29], expressed as:

\[ T_R = \frac{\pi R \sqrt{\rho G}}{0.1631 \nu + 0.8766} \]  

(C-11)

For simulations involving high particle coordination (4 or more), \( \delta t \approx 0.2T_R \) is suitable, whereas it can be raised to \( \delta t \approx 0.4T_R \) for lower coordination numbers (2 or less).

Once the simulation ends, the user can resort to the Analyst module to post-process the simulation. Data such as particle number, positions or velocities, efforts of the granular material on geometries or compressive forces can be extracted through a sophisticated exportation query tool. Customized properties can even be defined in the API, such as the total energy spent by the wheel, as defined by the integral over the time of the product between the external torque applied on the wheel and the rotational velocity of the wheel (cf. Sec. 4.1.2).
Appendix D: Observational error of the angle of repose simulations

The relatively small number of particles used for the simulations (3,000) raises the issue of the accuracy and the repeatability of the measure. The particles are generated randomly in the tube at the beginning of each simulation and the angle of repose measured is the mean of the slope angle of several fittings, as explained in Sec. 3.4.3. These two factors introduce an error in the measure. Therefore, an estimate of its precision is needed.

First of all, the observational error of the angle of repose was assumed to follow a normal distribution about its true value $A$. This means that the probability of measuring value $x$ is $p(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp(-\frac{(x-A)^2}{2\sigma^2})$, $\sigma$ being the standard deviation of the normal distribution.

If $n$ simulations are run, giving $n$ measures $(x_1, x_2, \ldots x_n)$, the estimators for the mean $A$ and the standard deviation $\sigma$ are

$$\hat{A} = \frac{1}{n} \sum_{i=1}^{n} x_i , \quad \hat{\sigma} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{A})^2} \quad (D-12)$$

To get a confidence interval, Bienaymé Chebyshev inequality is used:

$$P \left( |\hat{A} - A| > \epsilon \right) \leq \frac{V(\hat{A})}{\epsilon^2} , \quad \forall \epsilon > 0 \quad (D-13)$$

The variance of $\hat{A}$ around its true value $A$ being $V(\hat{A}) = \frac{\sigma^2}{n}$, it can be estimated by $V(\hat{A}) = \frac{\hat{\sigma}^2}{n}$. Eq. (D-13) becomes

$$P \left( |\hat{A} - A| > \epsilon \right) \leq \frac{\hat{\sigma}^2}{n\epsilon^2} , \quad \forall \epsilon > 0 \quad (D-14)$$

Using $\epsilon = \frac{\hat{\sigma}}{\sqrt{0.1n}}$ in Eq. (D-14), the 90% confidence interval is then given by

$$A \in \left[ \hat{A} - \frac{\hat{\sigma}}{\sqrt{0.1n}} , \hat{A} + \frac{\hat{\sigma}}{\sqrt{0.1n}} \right] \quad (D-15)$$
Five identical simulations are run, with the following interaction parameters: $e = 0.5, \mu_s = 0.3, \mu_r = 0.1$. The measured angles of repose were 26.80°, 27.95°, 26.93°, 27.57° and 28.09°, which gives estimated angle of repose and standard deviation of $\hat{A} = 27.47°$ and $\hat{\sigma} = 0.525°$ respectively. The 90% confidence interval is then [26.73°, 28.21°], or $27.47° \pm 0.74°$. Consequently, the real angle of repose is an element of this interval, with a probability of 90%. Another set of five identical simulations was run for another set of interaction parameters ($e = 0.5, \mu_s = 1.3, \mu_r = 0$), giving the following estimated angle of repose and standard deviation: $\hat{A} = 18.11°$ and $\hat{\sigma} = 0.167°$.

As the computing resource is limited and each simulation takes hours, the repose of a pile of particles is simulated only once per design variable set ($n = 1$), and not five times ($n = 5$) as described in the previous paragraph. This measure will then have an observational error. The ratio of the standard deviation over angle of repose is assumed to be lower than 0.02 (in the two previous cases, this ratio was $\sigma/A = 0.019$ and $\sigma/A = 0.0092$). Then, the interval of error of any angle of repose $\hat{A}$ measured on a single simulation is given by

$$ A \in \left[ \hat{A} - \frac{0.02\hat{A}}{\sqrt{0.1}}, \hat{A} + \frac{0.02\hat{A}}{\sqrt{0.1}} \right] $$

or in other words: $A = \hat{A} \pm 6.32%$. Consequently, it is assumed in this paper that the angle of repose measured on any simulation run only once has an observational error of less than 7%.
Appendix E: Results of direct shear test and angle of repose simulations

Table E-1: Results of the angle of repose simulations, involving 3,000 spherical particles.

<table>
<thead>
<tr>
<th>$r_0$</th>
<th>$\mu_r$</th>
<th>$\mu_s$</th>
<th>$A$</th>
<th>$r_0$</th>
<th>$\mu_r$</th>
<th>$\mu_s$</th>
<th>$A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>50$\mu$m</td>
<td>0.0</td>
<td>11.13$^\circ$</td>
<td>0.2</td>
<td>0.9</td>
<td>39.53$^\circ$</td>
<td>0.1</td>
<td>15.09$^\circ$</td>
</tr>
</tbody>
</table>
Table E-2: Results of the direct shear test simulations with spherical particles and various particle mean size.

<table>
<thead>
<tr>
<th>$r_0$</th>
<th>$\mu_r$</th>
<th>$\mu_s$</th>
<th>$c$ [kPa]</th>
<th>$\varphi$</th>
<th>$r_0$</th>
<th>$\mu_r$</th>
<th>$\mu_s$</th>
<th>$c$ [kPa]</th>
<th>$\varphi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>50µm</td>
<td>0</td>
<td>0.1</td>
<td>2.01</td>
<td>19.01°</td>
<td>0.2</td>
<td>0.3</td>
<td>13.4</td>
<td>14.84°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.3</td>
<td>3.02</td>
<td>30.60°</td>
<td>0.2</td>
<td>0.5</td>
<td>12.7</td>
<td>21.77°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.5</td>
<td>-3.52</td>
<td>36.91°</td>
<td>0.2</td>
<td>0.9</td>
<td>8.38</td>
<td>32.83°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.9</td>
<td>-11.6</td>
<td>44.07°</td>
<td>0.3</td>
<td>0.3</td>
<td>14.7</td>
<td>8.87°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1.3</td>
<td>-13.1</td>
<td>45.13°</td>
<td>0.3</td>
<td>0.5</td>
<td>18.8</td>
<td>14.28°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.1</td>
<td>4.35</td>
<td>16.41°</td>
<td>0.3</td>
<td>0.9</td>
<td>19.7</td>
<td>26.48°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.3</td>
<td>0.258</td>
<td>29.99°</td>
<td>0.05</td>
<td>0.5</td>
<td>-15.24</td>
<td>31.03°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.5</td>
<td>-3.798</td>
<td>33.86°</td>
<td>0.1</td>
<td>0.3</td>
<td>-8.89</td>
<td>25.38°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.9</td>
<td>-2.69</td>
<td>40.55°</td>
<td>0.05</td>
<td>0.5</td>
<td>3.215</td>
<td>32.83°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.1</td>
<td>8.94</td>
<td>7.31°</td>
<td>0.1</td>
<td>0.3</td>
<td>4.418</td>
<td>24.12°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.3</td>
<td>3.69</td>
<td>25.61°</td>
<td>0.05</td>
<td>0.5</td>
<td>0.888</td>
<td>27.23°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.5</td>
<td>3.26</td>
<td>30.1°</td>
<td>0.1</td>
<td>0.3</td>
<td>8.84</td>
<td>21.6°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.9</td>
<td>-4.05</td>
<td>39.41°</td>
<td>0.1</td>
<td>0.3</td>
<td>8.84</td>
<td>21.6°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1.3</td>
<td>-6.436</td>
<td>43.66°</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

130
Appendix F: MATLAB function used to compute the angle of repose of a pile of particles

Sec. 3.4.3 details how the angle of repose of a pile of 3,000 particles was computed from a 3 column table containing their coordinates at the end of the simulation. Here is the complete MATLAB code used:

```matlab
function [AoR] = angle_of_repose(filename)
% This function takes as an argument the name of the file containing the coordinates
% of the particles.
% Number of planes in which the angle of repose should be computed (the
% angle of repose is computed on both sides of the plane):
N = 12;
% Initialization of the table in which the angle of repose of each
% planes will be stored:
Result = zeros(2*N,1);
% Maximum radius of particles, in millimeters:
r = 0.075;
% The angle of the slope at the top of the pile is lower than the real
% angle of repose of the pile. We will then ignore the particles closer
% than D times the radius r of the axis of the pile. Value for D:
D = 3;

% Extraction of data:
% The data (position X Y Z) of the particles must be extracted with EDEM,
% using the following settings:
% Export format: text
% Line break at 1 column
% Delimiter: ' ' (space)
% Queries: Q1: position X (cm)
% Q2: position Y (cm)
% Q3: position Z (cm)

% The file compiled by EDEM and containing the data needs to be reformated
% to be easily used in our program.
% Let us form the three-column matrix 'part' that will contain the position
% of every particle, by reading the EDEM file line by line:
part = [0];
data = fopen(filename);
line = fgetl(data);
while ~isequal(line, 'EXTRACTED DATA')
    line = fgetl(data);
end
line = fgetl(data);
line = fgetl(data);
line = fgetl(data);
i = 1;
while line(1) ~= 'Q'
    part(i,1) = str2num(line);
end
```

131
i = i +1; 
line = fgets (data);

N_part = size (part); 
N_part = N_part (1,1); 
line = fgets (data); 
i = 1; 
while i<(N_part+1) 
part (i,2) = str2num(line); 
i = i +1; 
line = fgets (data); 
end 
line = fgets (data); 
i = 1; 
while i<(N_part+1) 
part (i,3) = str2num(line); 
i = i +1; 
line = fgets (data); 
end 
ang = 0; 

% Now we analyse the data on each one of the N planes. For each plane, two % txt files containing the position of the particles of the surface of the % pile are created. Then the angle of repose if obtained using the Matlab % fitting tools.

for i=1:N 
  % Initialisation of the arrays that will store the particle % positions in order to compute the angle of repose: 
data_pos = [0 0]; 
data_neg = [0 0]; 
  for j=1:N_part 
    % Let us compute its coordinates in the coordinate system which % is a rotation from the XYZ of and angle 'ang': 
x = cos(ang)*part(j,1) + sin(ang)*part(j,2); 
y = cos(ang)*part(j,2) - sin(ang)*part(j,1); 
z = part(j,3); 
    % The distance of the particle from the axis is: 
rho = sqrt(x^2 + y^2); 
    % Several conditions need to be satisfied to take this particle % into account in the fitting process to get the angle of % repose: 
    % - it has to be far enough from the axis of the pile (further % than Dw). 
    % - it has to be close enough to the considered plane (closer % than one radius r) 
    % - its altitude has to be higher than r, otherwise it is % lying on the ground so it might not be in the pile. 
    % - it has to be on the surface of the pile. 
    % Let us first check the 3 first ones 
if ( (rho>=Dw) ) & (y<=r) & (z>=r) 
  % Then , if z is positive, the particle might be % added in data_pos, else it might be added in data_neg. 
  % The 'data' matrix are constructed as follow : the row % k of the matrix contains the coordinates (x,z) of the % highest particle which satisfies (D + (k-1)*n)*r < x < % (D + k*n)*r. n*r is the length of each interval, in which % only the highest particle is taken into account. 
  n = 1.7; 
  if (x > 0) The particle might be added in data_pos

132
% Interval in which the particle \( j \) belongs:
\[
\text{int}_j = \lfloor \frac{x}{(n*r)-D/n+1} \rfloor;
\]
if \( \text{int}_j < 1 \)
\[
\text{int}_j = 1;
\]
end
% The particle is added if it is the highest in its
% interval:
if \( \text{size}(\text{data}_{\text{pos}},{1}) < \text{int}_j \)
\[
\text{data}_{\text{pos}}(\text{int}_j,1)=x;
\]
\[
\text{data}_{\text{pos}}(\text{int}_j,2)=z;
\]
else
\[
\text{if} (\text{data}_{\text{pos}}(\text{int}_j,2)<z)
\]
\[
\text{data}_{\text{pos}}(\text{int}_j,1)=x;
\]
\[
\text{data}_{\text{pos}}(\text{int}_j,2)=z;
\]
end
end
else
% The particle might be added in \( \text{data}_{\text{neg}} \):
% Interval in which the particle \( j \) belongs:
\[
\text{int}_j = \lfloor \frac{\text{abs}(x)}{(n*r)-D/n+1} \rfloor;
\]
if \( \text{int}_j < 1 \)
\[
\text{int}_j = 1;
\]
end
% The particle is added if it is the highest in its
% interval:
if \( \text{size}(\text{data}_{\text{neg}},{1}) < \text{int}_j \)
\[
\text{data}_{\text{neg}}(\text{int}_j,1)=x;
\]
\[
\text{data}_{\text{neg}}(\text{int}_j,2)=z;
\]
else
\[
\text{if} (\text{data}_{\text{neg}}(\text{int}_j,2)<z)
\]
\[
\text{data}_{\text{neg}}(\text{int}_j,1)=x;
\]
\[
\text{data}_{\text{neg}}(\text{int}_j,2)=z;
\]
end
end
end
% \( \text{data}_{\text{pos}} \) and \( \text{data}_{\text{neg}} \) might have empty intervals. If we don’t
% remove them, they will be seen as particles in position 0,0
% by the fitting algorithm. Let us check and remove any empty
% intervals now:
\[
\text{start} = 1;
\]
while \( \text{min}(\text{data}_{\text{pos}}(:,2))==0 \)
\[
\text{if} \text{data}_{\text{pos}}(\text{start},2)==0
\]
\[
\text{if} \text{start} == 1
\]
\[
\text{data}_{\text{pos}} = \text{data}_{\text{pos}}(2:\text{size}(\text{data}_{\text{pos}},{1}),1:2);
\]
\[
\text{else}
\]
\[
\text{data}_{\text{pos}} = [\text{data}_{\text{pos}}(1:\text{start}-1,1:2); \text{data}_{\text{pos}}(\text{start}+1:\text{size}(\text{data}_{\text{pos}},{1}),1:2)];
\]
\[
\text{end}
\]
\[
\text{else}
\]
\[
\text{start} = \text{start} + 1;
\]
\[
\text{end}
\]
end
\[
\text{start} = 1;
\]
while \( \text{min}(\text{data}_{\text{neg}}(:,2))==0 \)
\[
\text{if} \text{data}_{\text{neg}}(\text{start},2)==0
\]
\[
\text{if} \text{start} == 1
\]
\[
\text{end}
\],
\[
\text{end}
\]
\[
\text{end}
\]
\[
133
\]
data_neg = data_neg(2:size(data_neg,1),1:2);
else
    data_neg = [data_neg(1:start-1, 1:2); data_neg(start+1:size(data_neg,1), 1:2)];
end
else
    start = start + 1;
end

% Compute the linear least-square fit for data_pos and data_neg and
% write it in the results vector 'Result' :
a = polyfit(data_pos(:,1), data_pos(:,2), 1);
Result(i) = abs(atan(a(1,1))/pi*180);
a = polyfit(data_neg(:,1), data_neg(:,2), 1);
Result(i+N) = (atan(a(1,1))/pi*180);

% Update of "ang" value for next plane :
ang = ang + pi/N;
end

AoR = mean(Result)
end
Appendix G: Justification of the single sphere particle prototype with angle of repose simulations

The macroscopic properties of granular soils such as lunar regolith or sand highly depend on the shape and surface roughness of the soil grains. When it comes to modelling these soils in a DEM environment, a trade-off has to be found between particle shape fidelity and simulation runtime. Indeed, more complex shapes can be achieved by using multi-sphere particles, but it lengthens the simulation time by increasing the number of spheres in the system. This appendix shows that the angle of repose of a multi-sphere particle soil could be modelled by a single-sphere particle one with the introduction of an artificial rolling friction coefficient, described in Sec. 2.2.2. This shows that using multi-sphere particles is not necessary to illustrate the effect of surface roughness, justifying the single-sphere particle model used in this study.

As detailed in Sec. 3.4.3, Ji et al. modelled the angle of repose experiment in 2D DEM simulations [12]. Ji et al. compared the angle of repose for soils consisting of circular particles on one hand, and particles consisting of paired overlapping circles on the other hand, concluding that the paired particles have a higher angle of repose due to a higher friction between contacting particles, as shown in Fig. G-9.

Similar results were found in 3D simulations using particles made of two overlapping spheres. When such paired particles are used, the angle of repose increases with the particle aspect ratio, as shown in Fig. G-10. It was found that increasing the aspect ratio of paired particles had the same consequences on the angle of repose than using single-spheres particles with increasing rolling friction coefficient $\mu_r$ (cf. Fig. 3–12).

As a consequence, the use of spherical particles was chosen to model the soil, decreasing by a factor of two the number of spheres in the simulation system, and thus the
Figure G-9: 2D DEM angle of repose simulations of circular and paired (clumped) particles on Earth and Moon gravities as a function of static friction coefficient, by S. Ji et al. [12].

Figure G-10: 3D DEM angle of repose simulations of paired particles with various aspect ratios, as a function of static friction coefficient.
simulation runtime, while keeping a wide spectrum of possible angle of repose values in the soil calibration.
Appendix H: MATLAB function used to build a geometry-wheel made of an assembly of triangles.

In Sec. 4.1.1, the wheel parametrization and construction is described. Here is the MATLAB function used to build the XML code describing the wheel nodes and triangles.

```matlab
function [ ] = CAD_to_XML( r , w, N, alpha , beta0 , h0 , wt0 , wb0 , beta1 , h1 , wt1 , wb1 )

% This function will build the CAD model of a wheel as a function of the
% input parameters:
% r : wheel radius.
% w : wheel width
% N : number of grousers (99 maximum)
% alpha : angle of the V-shaped grousers (degrees)
% beta0 : grouser’s wall angle to the wheel surface (0=orthogonal) (degrees)
% h0 : grouser’s height
% wt0 : grouser’s width at top
% wb0 : grouser’s width at base
% 0 refers to the middle cross-section of the grouser, 1 to the side
% cross-section.
% ALL VALUES IN METERS and degrees !!!!!!

% the name of the xml file created will be :
name = [ 'CADwheel_Rad' , num2str(r) , '_N' , num2str(N) , '_h0' , num2str(h0) , '_alpha' , num2str(alpha) , '.xml'] ;

% conversion of the angles into radians :
alpha = alpha / 180 ;
beta0 = beta0 / 180 ;
beta1 = beta1 / 180 ;

% We will write the nodes and triangles in 2 separate files
% (the gain and load of the servocontrol have to be specified here.)
% Specify here the wheel dynamics :
dlmwrite(name , '<dynamics>' , 'append' , 'delimiter' , ' ');
dlmwrite(name , '<linear_translation start_vel_x =0' , 'start_vel_y =0' , 'start_vel_z =0' , 't_start =1' , 't_end =10' , 'accel_x =0' , 'accel_y =0' , 'accel_z =0' , 'name ="translation" start_vel_x =0.03' , 'delimiter' , ' ');%

138
```

138
% Let us verify that there is not too many grousers (the distance between two grousers must be positive):

circumference = 2*pi*r;
if N*wbl > circumference
    N = floor(circumference/wbl);
end

%Sooil height in the bin: (altitude of the highest soil particles at rest)
soil_height = 0.026;

% The wheel is built in the XZ plane, centered at the position:
x_0 = r + max(h0, h1) + 0.02; % wheel at 2cm from the left wall of the bin
y_0 = 0;
z_0 = soil_height + max(h0, h1)+r+0.002; % wheel at 0.2 cm of the soil
center = [x_0; y_0; z_0];

% The node and triangle id will start from 10.000 to avoid any interference
% with nodes and triangles of the existing geometries (bin, ...)
id_n = 10000;
id_t = 10000;

% If a previous version of the file exists, it is deleted:
if exist('nodes.xml')~=0
    delete('nodes.xml');
end

% Let us set up the header of the wheel part in the xml document, setting
% the translation, rotation and servoloads:

dlmwrite('nodes.xml', '<group starting_charge="0" material="aluminium" gain="1e-6" desired_force_x="0" desired_force_y="0" desired_force_z="2" dynamics_time="0" closure="volume" type="real" id="2" exclude_from_electrostatics="0" shape="default" name="Wheel" start/>' , '−append', 'delimiter', '');
dlmwrite('nodes.xml', '<dynamics>' , '−append', 'delimiter', '');
dlmwrite('nodes.xml', '<linear_translation start_vel_y="0" start_vel_z="0" start_time="0.25" name="Translation" start_vel_x="0.03"/>' , '−append', 'delimiter', '');
dlmwrite('nodes.xml', strcat('<linear_rotation moves_with_body="yes" ang_accn_y="0" start_angvel_z="0" ang_accn_z="0" start_time="0.25" origin_x="" origin_y="" origin_z="" num2str(center(1)) , num2str(center(2)) , num2str(center(3)) ," name="Rotation" start_angvel_x="0" ang_accn_x="0" start_angvel_y="/'' ,' −append', 'delimiter', '');
dlmwrite('nodes.xml', '</dynamics>', '−append', 'delimiter', '');

% If N==0: we build a flat wheel:
if N==0
% construction of flat wheel here.
% ...
% Test of construction for flat wheel:
% CAD to XML (0.0675, 0.004, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

% Writing the nodes:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 
% Writing the nodes: 
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 

dlmwrite('nodes.xml','<nodes>','-append','delimiter','');

% generic line to write a node:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %
% Writing the nodes: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 

% Two nodes at the center of the wheel sides:
% node #9999 is the center of front side, node #9998 is the center of back side:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %
% Writing the nodes: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 

dlmwrite('nodes.xml',strcat('<node x="',num2str(x0),'", y="',num2str(y0-w/2),'", z="',num2str(z0),'", id="',num2str(9999),'",/>
');','-append','delimiter','');
dlmwrite('nodes.xml',strcat('<node x="',num2str(x0),'", y="',num2str(y0+w/2),'", z="',num2str(z0),'", id="',num2str(9998),'",/>
');','-append','delimiter','');

% 36 nodes along the circumference of each side:
% on front side, nodes are labelled 10001 to 10036, on back 10101 to 10136
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %
% Writing the nodes: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 

node_front = [x0; y0-w/2; (z0+r)];
dlmwrite('nodes.xml',strcat('<node x="',num2str(node_front(1)),'", y="',num2str(node_front(2)),'", z="',num2str(node_front(3)),'", id="',num2str(10001),'",/>
');','-append','delimiter','');

node_back = [x0; y0+w/2; (z0+r)];
dlmwrite('nodes.xml',strcat('<node x="',num2str(node_back(1)),'", y="',num2str(node_back(2)),'", z="',num2str(node_back(3)),'", id="',num2str(10101),'",/>
');','-append','delimiter','');

for i=1:35
    point_front = rotation(node_front, center, i/36*2*pi);
    point_back = rotation(node_back, center, i/36*2*pi);
    dlmwrite('nodes.xml',strcat('<node x="',num2str(point_front(1)),'", y="',num2str(point_front(2)),'", z="',num2str(point_front(3)),'", id="',num2str(10001+i),'",/>
');','-append','delimiter','');
    dlmwrite('nodes.xml',strcat('<node x="',num2str(point_back(1)),'", y="',num2str(point_back(2)),'", z="',num2str(point_back(3)),'", id="',num2str(10101+i),'",/>
');','-append','delimiter','');
    i = i+1;
end

dlmwrite('nodes.xml','</nodes>','-append','delimiter','');
%Front side and back side:
for i = 1:35
    dlmwrite('nodes.xml', strcat('<triangle id=', num2str(i), ' node2=', num2str(10000+i), ' node3=', num2str(1001+i), '/>'), '-append', 'delimiter', '');
    dlmwrite('nodes.xml', strcat('<triangle id=', num2str(i+36), ' node2=', num2str(10100+i), ' node3=', num2str(10101+i), '/>'), '-append', 'delimiter', '');
end
dlmwrite('nodes.xml', '<triangle id=36 node1=9999 node2=10001 node3=10136 />', '-append', 'delimiter', ' ');
dlmwrite('nodes.xml', '<triangle id=72 node1=9998 node2=10101 node3=10136 />', '-append', 'delimiter', ' ');

%Tread:
for i = 1:35
    dlmwrite('nodes.xml', strcat('<triangle id=', num2str(100+i), ' node2=', num2str(10000+i), ' node3=', num2str(10100+i), '/>'), '-append', 'delimiter', ' ');
    dlmwrite('nodes.xml', strcat('<triangle id=', num2str(200+i), ' node2=', num2str(i), ' node3=', num2str(i), '/>'), '-append', 'delimiter', ' ');
end
dlmwrite('nodes.xml', '<triangle id=136 node1=10036 node2=10001 node3=10136 />', '-append', 'delimiter', ' ');
dlmwrite('nodes.xml', '<triangle id=236 node1=10136 node2=10101 node3=10001 />', '-append', 'delimiter', ' ');

% If N!=0 : we build the groused wheel:
else
    fprintf('Position of nodes of grouser 1.
');
    %Writing the nodes:
    % Two nodes at the center of the wheel sides: node #9999 is the center of front side, node #9998 is the center of back side:
    dlmwrite('nodes.xml', '<node x=', num2str(x0), ' y=', num2str(y0 - w/2), ' z=', num2str(z0), '/>','id=9999''/>' , '-append', 'delimiter', ' ');
    dlmwrite('nodes.xml', '<node x=', num2str(x0), ' y=', num2str(y0 + w/2), ' z=', num2str(z0), '/>','id=9998''/>' , '-append', 'delimiter', ' ');
    % One at the center of middle cross section:
    dlmwrite('nodes.xml', '<node x=', num2str(x0), ' y=', num2str(y0), ' z=', num2str(z0), '/>','id=9997''/>' , '-append', 'delimiter', ' ');

    fprintf('Coordinates of the points of the first grouser : node20101 = [x0; y0; (z0 + r)];
    mu = asin(w*tan(alpha)/(2*r));
    node_20101 = rotation(node_20101, center, -mu) + [0; -w/2; 0];
    node_30101 = rotation(node_20101, center, -mu) - [0; -w/2; 0];
end
\[
gamma = 2 \cdot \arcsin \left( \frac{w_0}{2 \cdot r} \right);
\]

\[
\lambda_0 = 2 \cdot \arcsin \left( \frac{w_0}{2 \cdot (r+h_0)} \right);
\]

\[
\lambda_1 = 2 \cdot \arcsin \left( \frac{w_1}{2 \cdot (r+h_1)} \right);
\]

\[
\rho_0 = 2 \cdot \frac{w_0}{r};
\]

\[
\rho_1 = 2 \cdot \frac{w_1}{r};
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_20101(1)), '", i="', num2str(node_20101(2)), '", i="', num2str(node_20101(3)), '", id="20101"/>'), '');
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_10101(1)), '", i="', num2str(node_10101(2)), '", i="', num2str(node_10101(3)), '", id="10101"/>'), '');
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_30101(1)), '", i="', num2str(node_30101(2)), '", i="', num2str(node_30101(3)), '", id="30101"/>'), '');
\]

\[
\text{node_20401 = rotation( node_20101, center, (-gamma));
\]

\[
\text{node_10401 = rotation( node_10101, center, (-gamma));
\]

\[
\text{node_30401 = rotation( node_30101, center, (-gamma));
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_20401(1)), '", i="', num2str(node_20401(2)), '", i="', num2str(node_20401(3)), '", id="20401"/>'), '');
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_10401(1)), '", i="', num2str(node_10401(2)), '", i="', num2str(node_10401(3)), '", id="10401"/>'), '');
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_30401(1)), '", i="', num2str(node_30401(2)), '", i="', num2str(node_30401(3)), '", id="30401"/>'), '');
\]

\[
\text{node_20201 = node_20101+(-h_0*\sin(beta_0));
\]

\[
\text{node_10201 = node_10101+(-h_1*\sin(beta_1+mu));
\]

\[
\text{node_30201 = node_30101+(-h_1*\sin(beta_1+mu));
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_20201(1)), '", i="', num2str(node_20201(2)), '", i="', num2str(node_20201(3)), '", id="20201"/>'), '');
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_10201(1)), '", i="', num2str(node_10201(2)), '", i="', num2str(node_10201(3)), '", id="10201"/>'), '');
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_30201(1)), '", i="', num2str(node_30201(2)), '", i="', num2str(node_30201(3)), '", id="30201"/>'), '');
\]

\[
\text{lambda_0 = 2 \cdot \arcsin \left( \frac{w_0}{2 \cdot (r+h_0)} \right);
\]

\[
\text{node_20301 = rotation( node_20201, center, (-lambda_0));
\]

\[
\text{lambda_1 = 2 \cdot \arcsin \left( \frac{w_1}{2 \cdot (r+h_1)} \right);
\]

\[
\text{node_10301 = rotation( node_10201, center, (-lambda_1));
\]

\[
\text{node_30301 = rotation( node_30201, center, (-lambda_1));
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_20301(1)), '", i="', num2str(node_20301(2)), '", i="', num2str(node_20301(3)), '", id="20301"/>'), '');
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_10301(1)), '", i="', num2str(node_10301(2)), '", i="', num2str(node_10301(3)), '", id="10301"/>'), '');
\]

\[
\text{dlmwrite('nodes.xml', strcat('<node i="', num2str(node_30301(1)), '", i="', num2str(node_30301(2)), '", i="', num2str(node_30301(3)), '", id="30301"/>'), '');
\]

\[
\text{rho_0 = 2 \cdot p_1/N - w_0/r;
\]

\[
\text{node_20501 = rotation( node_20101, center, (rho_0/2));
\]

\[
\text{rho_1 = 2 \cdot p_1/N - w_1/r;
\]
node_10501 = rotation( node_10101, center, (rho1/2));
node_30501 = rotation( node_30101, center, (rho1/2));
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_20501(1)), ''' y=''', num2str
( node_20501(2)), ''' z=''', num2str(node_20501(3)), ''' id=''', num2str
( node_20501(4)), ''' delimeter ''');
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_10501(1)), ''' y=''', num2str
( node_10501(2)), ''' z=''', num2str(node_10501(3)), ''' id=''', num2str
( node_10501(4)), ''' delimeter ''');
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_30501(1)), ''' y=''', num2str
( node_30501(2)), ''' z=''', num2str(node_30501(3)), ''' id=''', num2str
( node_30501(4)), ''' delimeter ''');

%Let's write the other nodes, for the next N-1 groups:

for i=1:(N-1)

fprintf( strcat( 'Position of_nodes_of_grouser-', num2str(i+1), '.\n'));
node_1010k = rotation( node_10101, center, i/(N-1)*2*pi);
node_2010k = rotation( node_20101, center, i/(N-1)*2*pi);
node_3010k = rotation( node_30101, center, i/(N-1)*2*pi);
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_1010k(1)), ''' y=''', num2str
( node_1010k(2)), ''' z=''', num2str(node_1010k(3)), ''' id=''', num2str
( node_1010k(4)), ''' delimeter ''');
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_2010k(1)), ''' y=''', num2str
( node_2010k(2)), ''' z=''', num2str(node_2010k(3)), ''' id=''', num2str
( node_2010k(4)), ''' delimeter ''');
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_3010k(1)), ''' y=''', num2str
( node_3010k(2)), ''' z=''', num2str(node_3010k(3)), ''' id=''', num2str
( node_3010k(4)), ''' delimeter ''');

node_1020k = rotation( node_10201, center, i/(N-1)*2*pi);
node_2020k = rotation( node_20201, center, i/(N-1)*2*pi);
node_3020k = rotation( node_30201, center, i/(N-1)*2*pi);
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_1020k(1)), ''' y=''', num2str
( node_1020k(2)), ''' z=''', num2str(node_1020k(3)), ''' id=''', num2str
( node_1020k(4)), ''' delimeter ''');
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_2020k(1)), ''' y=''', num2str
( node_2020k(2)), ''' z=''', num2str(node_2020k(3)), ''' id=''', num2str
( node_2020k(4)), ''' delimeter ''');
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_3020k(1)), ''' y=''', num2str
( node_3020k(2)), ''' z=''', num2str(node_3020k(3)), ''' id=''', num2str
( node_3020k(4)), ''' delimeter ''');

node_1030k = rotation( node_10301, center, i/(N-1)*2*pi);
node_2030k = rotation( node_20301, center, i/(N-1)*2*pi);
node_3030k = rotation( node_30301, center, i/(N-1)*2*pi);
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_1030k(1)), ''' y=''', num2str
( node_1030k(2)), ''' z=''', num2str(node_1030k(3)), ''' id=''', num2str
( node_1030k(4)), ''' delimeter ''');
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_2030k(1)), ''' y=''', num2str
( node_2030k(2)), ''' z=''', num2str(node_2030k(3)), ''' id=''', num2str
( node_2030k(4)), ''' delimeter ''');
dlmwrite( 'nodes.xml', strcat( '<node x=''', num2str(node_3030k(1)), ''' y=''', num2str
( node_3030k(2)), ''' z=''', num2str(node_3030k(3)), ''' id=''', num2str
( node_3030k(4)), ''' delimeter ''');

node_1040k = rotation( node_10401, center, i/(N-1)*2*pi);
node_2040k = rotation( node_20401, center, i/(N-1)*2*pi);
node_3040k = rotation(node_30401, center, i/(N-1)+2*pi);
dlmwrite('nodes.xml', strcat ('<node,x="', num2str(node_2040k(1)), '" y="', num2str(node_2040k(2)), '" node1="9999" node2="9999" node3="9999" id="', num2str(20401+i), '"/>'), '-append', 'delimiter','');
dlmwrite('nodes.xml', strcat ('<node,x="', num2str(node_1040k(1)), '" y="', num2str(node_1040k(2)), '" node1="9999" node2="9999" node3="9999" id="', num2str(10401+i), '"/>'), '-append', 'delimiter','');
dlmwrite('nodes.xml', strcat ('<node,x="', num2str(node_3040k(1)), '" y="', num2str(node_3040k(2)), '" node1="9999" node2="9999" node3="9999" id="', num2str(30401+i), '"/>'), '-append', 'delimiter','');

node_1050k = rotation(node_10501, center, i/(N-1)+2*pi);
node_2050k = rotation(node_20501, center, i/(N-1)+2*pi);
node_3050k = rotation(node_30501, center, i/(N-1)+2*pi);
dlmwrite('nodes.xml', strcat ('<node,x="', num2str(node_2050k(1)), '" y="', num2str(node_2050k(2)), '" node1="9999" node2="9999" node3="9999" id="', num2str(20501+i), '"/>'), '-append', 'delimiter','');
dlmwrite('nodes.xml', strcat ('<node,x="', num2str(node_1050k(1)), '" y="', num2str(node_1050k(2)), '" node1="9999" node2="9999" node3="9999" id="', num2str(10501+i), '"/>'), '-append', 'delimiter','');
dlmwrite('nodes.xml', strcat ('<node,x="', num2str(node_3050k(1)), '" y="', num2str(node_3050k(2)), '" node1="9999" node2="9999" node3="9999" id="', num2str(30501+i), '"/>'), '-append', 'delimiter','');

end
dlmwrite('nodes.xml', '</nodes>', '-append', 'delimiter','');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Writing the triangles : %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

id = 1;

% Triangles of the two sides of the wheel :
fprintf('Triangles of the sides\n');
for i=1:(N-1)
    % Front side :
dlmwrite('nodes.xml', strcat ('<triangle id="', num2str(id), '" node1="9999" node2="9999" node3="9999" id="', num2str(10500+i), '"/>'), '-append', 'delimiter','');
    id = id +1;
dlmwrite('nodes.xml', strcat ('<triangle id="', num2str(id), '" node1="9999" node2="9999" node3="9999" id="', num2str(10400+i), '"/>'), '-append', 'delimiter','');
    id = id +1;
dlmwrite('nodes.xml', strcat ('<triangle id="', num2str(id), '" node1="9999" node2="9999" node3="9999" id="', num2str(10100+i), '"/>'), '-append', 'delimiter','');
    id = id +1;
dlmwrite('nodes.xml', strcat ('<triangle id="', num2str(id), '" node1="9999" node2="9999" node3="9999" id="', num2str(10000+i), '"/>'), '-append', 'delimiter','');
    id = id +1;

    % Back side :
dlmwrite('nodes.xml', strcat ('<triangle id="', num2str(id), '" node1="9999" node2="9999" node3="9999" id="', num2str(20500+i), '"/>'), '-append', 'delimiter','');
    id = id +1;
dlmwrite('nodes.xml', strcat ('<triangle id="', num2str(id), '" node1="9999" node2="9999" node3="9999" id="', num2str(20400+i), '"/>'), '-append', 'delimiter','');
    id = id +1;
dlmwrite('nodes.xml', strcat ('<triangle id="', num2str(id), '" node1="9999" node2="9999" node3="9999" id="', num2str(20100+i), '"/>'), '-append', 'delimiter','');
    id = id +1;
dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30100+i), 'node3=', num2str(30500+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30500+i), 'node3=', num2str(30400+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30400+i), 'node3=', num2str(30300+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30300+i), 'node3=', num2str(30200+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30200+i), 'node3=', num2str(30100+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30100+i), 'node3=', num2str(30400+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30400+i), 'node3=', num2str(30300+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30300+i), 'node3=', num2str(30200+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30200+i), 'node3=', num2str(30100+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30100+i), 'node3=', num2str(30400+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30400+i), 'node3=', num2str(30300+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30300+i), 'node3=', num2str(30200+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30200+i), 'node3=', num2str(30100+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30100+i), 'node3=', num2str(30400+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30400+i), 'node3=', num2str(30300+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30300+i), 'node3=', num2str(30200+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30200+i), 'node3=', num2str(30100+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(30100+i), 'node3=', num2str(30400+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

% Triangles of the tread:
fprintf(' Triangles of the tread \( g\) \( p\) \n');
for i=1:(N-1)
dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(10400+i), 'node3=', num2str(10500+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(10500+i), 'node3=', num2str(10300+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(10300+i), 'node3=', num2str(10200+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

dlmwrite('nodes.xml', strcat('<triangle id=', num2str(id), '>
', 'node2=', num2str(10200+i), 'node3=', num2str(10100+i), 'id=', id, 'delimiter', 'a'), 'id' = id + 1;

end
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(20300+i), ",node2="', num2str(30400+i), '",node3="', num2str(30300+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(10100+i), ",node2="', num2str(20100+i), '",node3="', num2str(20000+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(20200+i), ",node2="', num2str(20100+i), '",node3="', num2str(20000+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(20300+i), ",node2="', num2str(30400+i), '",node3="', num2str(30300+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(10200+i), ",node2="', num2str(20100+i), '",node3="', num2str(20000+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(20200+i), ",node2="', num2str(20100+i), '",node3="', num2str(20000+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(10300+i), ",node2="', num2str(20100+i), '",node3="', num2str(20000+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(20300+i), ",node2="', num2str(30400+i), '",node3="', num2str(30300+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(10500+i), ",node2="', num2str(20100+i), '",node3="', num2str(20000+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(20500+i), ",node2="', num2str(20100+i), '",node3="', num2str(20000+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(20400+i), ",node2="', num2str(30400+i), '",node3="', num2str(30300+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(20401+i), ",node2="', num2str(30401+i), '",node3="', num2str(30301+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
dlmwrite('nodes.xml', strcat('<triangle_id="', num2str(id), '",
num2str(20500+i), ",node2="', num2str(30401+i), '",node3="', num2str(30301+i), '",/>'), 'append', 'delimiter', '');
```
```matlab
fprintf(strcat('Triangles_of_the_tread(', grouper(n, num2str(i+1), ')') \n'));
```
end
```
dlmwrite('nodes.xml', strcat('<triangle id="', num2str(id), '" node1="', num2str(10500+N), '" node2="', num2str(20401), '" node3="', num2str(20500+N), '"/>',' delimiter ',''); id = id +1;

dlmwrite('nodes.xml', strcat('<triangle id="', num2str(id), '" node1="', num2str(20401), '" node2="', num2str(30401), '" node3="', num2str(20500+N), '"/>',' delimiter ',''); id = id +1;

dlmwrite('nodes.xml', strcat('<triangle id="', num2str(id), '" node1="', num2str(20500+N), '" node2="', num2str(30401), '" node3="', num2str(30500+N), '"/>',' delimiter ',''); id = id +1;

end

% The wheel is now built!

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Assembly of the final xml input deck:

% Delete file if it already exists:
if exist(name')~=0
    delete(name);
end

% copy the beginning, containing the soil description:
copyfile('xml_begin.xml',name)

    fprintf('Assembling the file. Copying the beginning.
');
data=fopen('nodes.xml');
line = fgetl(data);
while ischar(line)
    dlmwrite(name,line,'-append','delimiter','');
    line = fgetl(data);
end

% copy the end, containing some material properties:

    fprintf('Assembling the file. Copying the end.
');
data=fopen('xml_end.xml');
line = fgetl(data);
while ischar(line)
    dlmwrite(name,line,'-append','delimiter','');
    line = fgetl(data);
end
fclose('all');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The function CAD_to_XML used the subfunction rotation, used to compute the
% position of the grousers nodes:

function [ new_point ] = rotation( point, center, angle )

% Inputs:
% point : coordinates of the point we want to rotate [x, y, z]
% center : center of the rotation [x_0, y_0, z_0]
% angle : angle of rotation, in radian

% Output:
% new_point : coordinates of the rotated point.

% The rotation is on the Y axis.
rot_mat = [[\cos(\text{angle}) \ 0 \ \sin(\text{angle})];
            [0 \ 1 \ 0];
            [-\sin(\text{angle}) \ 0 \ \cos(\text{angle})]];

new_point = \text{rot_mat} \ast (\text{point} - \text{center}) + \text{center};

end
Appendix I: MATLAB function used to build a particle-wheel made of an assembly of spheres.

It is mentioned in Sec. 4.1.2 that in some wheel-soil simulations, the wheel actually consisted of a macro-particle made of an assembly of spheres. Here is the MATLAB function used to build the XML code describing the spheres forming the wheel.

```matlab
function [ ] = wheel_with_grousers( )

% This function will create an XML file with the positions of the
% overlapping spheres that will form the particle. This particle will
% represent a wheel with a given number of grousers. In the following, the
% length unit is meter. The wheel is in the (X,Z) plan, centered in 0,0.

% Let us first define the wheel parameters :
% Diameter of the wheel:
D = 0.10;
% Position of the wheel on the Y axis :
Y = 0;
y = num2str(Y);
% Number, heigth and thickness of grousers:
N_g = 36;
H_g = 0.008;
T_g = 0.004;

% Diameter of the spheres that make the particle (wheel):
d = 0.004;
r = d/2;
% Distance between the centers of two contacting spheres:
dist = 1.3*r;
% the name of the XML file created will be :
name = [ 'wheel_D_' num2str(D) 'T_' num2str(d) 'N_g_' num2str(N_g) 'H_g_' num2str(H_g) 'T_g_' num2str(T_g) '.xml' ];

% Then we compute the number of spheres needed for the wheel only (without grousers) :
N_s = round(pi*(D-d)/dist);

% Angle between directions of 2 contacting spheres:
ang = 2*pi/N_s;

id = 0;
% table of the centres of spheres:
```
CoordSpheres = [0,0];

% We create the spheres that make the shell of the wheel:
    for i = 0:(N_s-1)
        % Compute the coordinates of i-th wheel:
        x = (D/2-r)*cos(i*ang);
        z = (D/2-r)*sin(i*ang);
        S = ['<sphere x="', num2str(x), '" y="', num2str(z), '" contact_radius ="', num2str(r), '" id="', num2str(id), '" name="sphere', num2str(id), '" physical_radius="', num2str(r), '"/>'];
        id = id +1;
        CoordSpheres(id,1)=x;
        CoordSpheres(id,2)=z;
        dlmwrite(name, S, '-append', 'delimiter', '');
    end

% Let us now create the grousers:
    ang = 2*pi/N_g;
    for n = 0:N_g-1
        % If the grouser's thickness is smaller than the sphere's diameter, then the
        % grouser will be made of a single row of spheres, which radii will equal T_g/2:
        if T_g <= d
            r=T_g/2;
            % number of spheres per grouser:
            N_sg = round(H_g/r);
            for i = 1:N_sg
                x = (D/2 + H_g - i*r)*cos(n*ang);
                z = (D/2 + H_g - i*r)*sin(n*ang);
                S = ['<sphere x="', num2str(x), '" y="', num2str(z), '" contact_radius ="', num2str(r), '" id="', num2str(id), '" name="sphere', num2str(id), '" physical_radius="', num2str(r), '"/>'];
                id = id +1;
                CoordSpheres(id,1)=x;
                CoordSpheres(id,2)=z;
                dlmwrite(name, S, '-append', 'delimiter', '');
            end
        else, the radius of the spheres used to model the grousers remains r

        else % number of spheres on each sides of the grousers :
            N_sg = round(H_g/r);
            % number of spheres on the top side of the grousers :
            N_sg_top = round(T_g/r-2);
            % First we create the spheres on the sides :
            for i = 1:N_sg
                rho = sqrt((D/2 + H_g - i*r)^2 + (T_g/2 - r)^2);
                % % %
            end

        end

    end
\[ d_{\text{theta}} = \arcsin(\frac{T_g/2-r}{\rho}) \]

\[ x = \rho \cos(n \ast \text{ang} + d_{\text{theta}}) \]
\[ z = \rho \sin(n \ast \text{ang} + d_{\text{theta}}) \]

\[ S = ['<sphere' 'x=' \text{num2str}(x) ' 'y=' \text{num2str}(y) ' 'z=' \text{num2str}(z) ' 'contact\_radius=' \text{num2str}(r) ' 'id=' \text{num2str}(id) ' 'name='sphere' 'physical\_radius=' \text{num2str}(r) '>'] \]

id = id + 1;
CoordSpheres(id,1)=x;
CoordSpheres(id,2)=z;
dlmwrite(name,S,['-append','delimiter','','']);

\[ x = \rho \cos(n \ast \text{ang} - d_{\text{theta}}) \]
\[ z = \rho \sin(n \ast \text{ang} - d_{\text{theta}}) \]

\[ S = ['<sphere' 'x=' \text{num2str}(x) ' 'y=' \text{num2str}(y) ' 'z=' \text{num2str}(z) ' 'contact\_radius=' \text{num2str}(r) ' 'id=' \text{num2str}(id) ' 'name='sphere' 'physical\_radius=' \text{num2str}(r) '>'] \]

id = id + 1;
CoordSpheres(id,1)=x;
CoordSpheres(id,2)=z;
dlmwrite(name,S,['-append','delimiter','','']);

end

% then we create the top side of the grouser:

for i = 1:N_sg_top

\[ \rho = \sqrt{\left(\frac{D}{2} + H_g - r\right)^2 + \left(\frac{T_g}{2} - i \ast r\right)^2} \]
\[ d_{\text{theta}} = \arcsin(\frac{T_g/2-i \ast r}{\rho}) \]
\[ x = \rho \cos(n \ast \text{ang} + d_{\text{theta}}) \]
\[ z = \rho \sin(n \ast \text{ang} + d_{\text{theta}}) \]

\[ S = ['<sphere' 'x=' \text{num2str}(x) ' 'y=' \text{num2str}(y) ' 'z=' \text{num2str}(z) ' 'contact\_radius=' \text{num2str}(r) ' 'id=' \text{num2str}(id) ' 'name='sphere' 'physical\_radius=' \text{num2str}(r) '>'] \]

id = id + 1;
CoordSpheres(id,1)=x;
CoordSpheres(id,2)=z;
dlmwrite(name,S,['-append','delimiter','','']);

end

end
Appendix J: C++ plug-in used to apply external efforts on the particle wheels in simulations using the torque-controlled approach.

The salient parts of the C++ code of the plug-ins used in torque-controlled simulations (cf. Sec. 4.1.2) are given here. All of them are associated with a TXT file containing some parameters, referred to as param1, param2, . . . , in the following.

**PolyTorque**

```cpp
// At each timestep, the plug-in checks all particle types, and external forces are applied only on the 'wheel' particle:
if(strcmp(type, "wheel") == 0)
{
    // An external force on the Z direction is applied (weight on the wheel):
    calculatedForceX = 0;
    calculatedForceY = 0;
    calculatedForceZ = param1;

    // After a given time 'param2', an external torque in the Y axis is applied (torque of the motor on the wheel). This torque is modelled as a second order polynomial, function of the time.
    if (time < param2)
    {
        calculatedTorqueY = 0;
    }
    else
    {
        calculatedTorqueY = param3 + param4*time + param5*time*time;
    }
    calculatedTorqueX = 0;
    calculatedTorqueZ = 0;
}
return eSuccess;
```

**ServoSpeed**

```cpp
// At each timestep, the plug-in checks all particle types, and external forces are applied only on the 'wheel' particle:
if(strcmp(type, "wheel") == 0)
{
    // An external force on the Z direction is applied (weight on the wheel):
    calculatedForceX = 0;
```

153
calculatedForceY = 0;
calculatedForceZ = param1;

// After a given time 'param2', an external torque in the Y axis of value 'param3' is applied if the horizontal velocity of the wheel is lower than 'param4'. When the external torque is applied, a custom property, called EnergySpent and representing the total energy given by the torque to the wheel, is updated by adding the instant power (torque times angular velocity) multiplied by the simulation timestep during which the torque is applied.
if(time < param2)
{
    calculatedTorqueY =0;
}
else
{
    if(velX < param4)
    {
        calculatedTorqueY = param3;
        calculatedPropertyListChanges[m_energySpentOffset] += timestep*param3*angVelY;
    }
}
calculatedTorqueX = 0;
calculatedTorqueZ = 0;

return eSuccess;

ServoTorque

// At each timestep, the plug-in checks all particle types, and external forces are applied only on the 'wheel' particle:
if(strcmp(type, "wheel") ==0)
{
    // An external force on the Z direction is applied (weight on the wheel):
calculatedForceX = 0;
calculatedForceY = 0;
calculatedForceZ = param1;

    // After a given time 'param2', an external torque in the Y axis of value 'param3' is applied if the angular velocity of the wheel is lower than 'param4'.
    if(time < param2)
    {
        calculatedTorqueY =0;
    }
    else
    {
        if(angVelY < param4)
        calculatedTorqueY = param3;
    }
calculatedTorqueX = 0;
calculatedTorqueZ = 0;
}
return eSuccess;
Appendix K: The Newton-Gauss algorithm applied on an analytical example

Let us consider a simple optimization problem for which the solutions can be computed analytically in order to prove the efficiency of the Newton-Gauss algorithm. The following problem is considered:

\[
\begin{bmatrix}
 x_1 \\
 x_2 
\end{bmatrix}, \quad \phi(x) = \begin{bmatrix}
 \phi_1(x) = x_2 - x_1^2 \\
 \phi_2(x) = 1 - x_1 \\
 \phi_3(x) = -x_2 
\end{bmatrix} = \mathbf{0}.
\]  
(K-17)

This system is obviously overdetermined: the solution of the two first equations \( \phi_1(x) \) and \( \phi_2(x) \), \( x = [1, 1] \), is not a solution for \( \phi_3(x) \). If the same weight is applied to each \( \phi_i \), the function \( f \) will be:

\[
f(x) = \frac{1}{2} \phi^T \phi = \frac{1}{2} ((x_2 - x_1^2)^2 + (1 - x_1)^2 + (-x_2)^2)
\]
(K-18)

\[
= \frac{1}{2} (x_1^4 - 2x_1^2x_2 + x_1^2 + 2x_2^2 - 2x_1 + 1)
\]

The analytical solution \( x_0 \) minimizing \( f \) can be found by vanishing its Jacobian:

\[
x_0 = \left[ \left( \frac{1 + \sqrt{31}/27}{2} \right)^{\frac{1}{3}} + \left( \frac{1 - \sqrt{31}/27}{2} \right)^{\frac{1}{3}}, \frac{1}{2} \left( \left( \frac{1 + \sqrt{31}/27}{2} \right)^{\frac{1}{3}} + \left( \frac{1 - \sqrt{31}/27}{2} \right)^{\frac{1}{3}} \right)^2 \right]
\]  
(K-19)

or approximately \( x_0 \approx [0.6823, 0.2328] \), with \( f(x_0) \approx 0.1046 \), as it can be verified on the graph of \( f \) (cf. Fig. K-11).

The optimum value \( x_0 \) was then searched with the Newton-Gauss algorithm, and the values found were close to the analytical solution (cf. Tab. K-3).

The weighing matrix \( W \) is useful to give more or less importance to one of the \( \phi_i \).
function in the overdetermined system of Eq. [K-17]. For instance, if a diagonal matrix $W$ is used, with $w_i$ being the weighting factor of $\phi_i$ respectively, a smaller weighting factor $w_3$ will deviate the solution $x_0$ towards the solution $x = [1, 1]$ of the system

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \phi(x) = \begin{bmatrix} \phi_1(x) = x_2 - x_1^2 \\ \phi_2(x) = 1 - x_1 \end{bmatrix} = \mathbf{0} \quad (K-20)$$

as shown in Tab. K-4.
Table K-3: Example of Newton-Gauss algorithm: Optimum value $x_0$ given as a function of initial guess and tolerance (damping coefficient: $\alpha = 1$).

<table>
<thead>
<tr>
<th>Initial guess $x^0$</th>
<th>Tolerance $\epsilon$</th>
<th>Steps before convergence</th>
<th>Solution $x_0$</th>
<th>Value of $f(x_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0]</td>
<td>$10^{-1}$</td>
<td>3</td>
<td>0.6863, 0.2353</td>
<td>0.1080</td>
</tr>
<tr>
<td>[0, 0]</td>
<td>$10^{-2}$</td>
<td>4</td>
<td>0.6814, 0.2321</td>
<td>0.1047</td>
</tr>
<tr>
<td>[0, 0]</td>
<td>$10^{-4}$</td>
<td>7</td>
<td>0.6823, 0.2328</td>
<td>0.1046</td>
</tr>
<tr>
<td>[4, 7]</td>
<td>$10^{-1}$</td>
<td>4</td>
<td>0.6866, 0.2355</td>
<td>0.1077</td>
</tr>
<tr>
<td>[4, 7]</td>
<td>$10^{-2}$</td>
<td>5</td>
<td>0.6813, 0.2321</td>
<td>0.1047</td>
</tr>
<tr>
<td>[4, 7]</td>
<td>$10^{-4}$</td>
<td>8</td>
<td>0.6823, 0.2328</td>
<td>0.1046</td>
</tr>
<tr>
<td>[348, -253]</td>
<td>$10^{-1}$</td>
<td>7</td>
<td>0.6851, 0.2346</td>
<td>0.1092</td>
</tr>
<tr>
<td>[348, -253]</td>
<td>$10^{-2}$</td>
<td>8</td>
<td>0.6817, 0.2323</td>
<td>0.1047</td>
</tr>
<tr>
<td>[348, -253]</td>
<td>$10^{-6}$</td>
<td>14</td>
<td>0.6823, 0.2328</td>
<td>0.1046</td>
</tr>
</tbody>
</table>

Table K-4: Example of Newton-Gauss algorithm: Optimum value $x_0$, with function $\phi_3$ subjected to decreasing weighting factor (tolerance: $\epsilon = 10^{-5}$, damping coefficient: $\alpha = 1$, initial guess: $x^0 = [0, 0]$).

<table>
<thead>
<tr>
<th>Ratio $r = w_1/w_3 = w_2/w_3$</th>
<th>Solution $x_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.6823, 0.2328]</td>
</tr>
<tr>
<td>10</td>
<td>[0.8773, 0.6997]</td>
</tr>
<tr>
<td>100</td>
<td>[0.9812, 0.9532]</td>
</tr>
</tbody>
</table>

158
References


