

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Datum

Unterschrift

Preface of the author

I first realised my deep interest in numerical computations during my master thesis at the Institute of Mechanics at the Technical University of Graz. It was only through my former supervisor Dr. Mathias Mair that I became aware of the advertised doctoral position at the Institute of Soil Mechanics and Foundation Engineering.

The present doctoral thesis allowed me to deal with the numerical simulation methods in more detail. Moreover, the numerical investigation of a dynamic cone penetration test on Martian soil was an exciting application.

Therefore, I would like to thank my supervisor Univ.-Prof. Dipl.-Ing. Dr.techn. Helmut F. Schweiger M.Sc. of the Institute of Soil Mechanics and Foundation Engineering for giving me the opportunity to be one of his PhD students. I appreciate that he always took time for questions and discussions. I would also like to thank Univ.-Prof. Dipl.-Ing. Dr.techn. Roman Marte, the head of the institute, for the helpful discussions at the institute meetings.

In addition, I would like to thank Dr. Mag. Ing. Günter Kargl and Univ.-Doz. Dr. Norbert I. Kömle of the Space Research Institute in Graz for the support and the kind collaboration during my PhD. I would, furthermore, like to thank Dr. Mark S. Bentley for all the time he spends on helping me.

I am grateful to the Austrian FFG (Forschungs-Förderungs-Gesellschaft) for supporting this research in the frame of its ASAP10-program under the Project InSight-MPS (Modeling of Dynamic Penetration in Granular Soils under Space Conditions).

Also, I would like to thank my wife who moved with me to Graz and stand by me during my dissertation.

Graz, November 2017

Joshua Poganski

Kurzfassung

Numerische Modellierung einer Dynamischen Drucksondierung auf dem Mars

Die NASA InSight Mission ist eine unbemannte Raumfahrtmission, bei der ein stationärer Lander auf die Oberfläche des Mars geschickt wird um das Innere des Planeten zu untersuchen. Der geplante Start der Raumfahrtmission im Jahr 2016 musste aufgrund technischer Probleme auf das Jahr 2018 verschoben werden. Das Ziel der Mission ist die Untersuchung des inneren Aufbaus des Planeten durch die Beobachtung der seismischen Aktivität und des Wärmeflusses im Inneren des Planeten. Dazu wird ein Seismometer (SEIS) auf die Oberfläche platziert und ein Wärmeflussensor (HP³) in den Boden gerammt. Der Wärmeflussensor wird 3 bis 5 Meter tief in den Marsboden eingebracht um den thermischen Einfluss der Sonneneinstrahlung auf die Temperaturmessung zu verringern und um den Temperaturgradienten im Boden durch mehrere Temperatursensoren auf dem Schleppkabel in unterschiedlichen Tiefen zu bestimmen.

HP³ besitzt einen inneren Schlagmechanismus welcher das Instrument in den Boden rammt. Die Bewegung des Instruments wird während der Sondierung gemessen, um aus diesen Daten mechanische Eigenschaften des Marsbodens zu bestimmen. Das Wissen über die Bodenbeschaffenheit und deren bodenmechanischen Eigenschaften liefert Rückschlüsse über die Geschichte des Planeten und soll zukünftigen Missionen helfen, diese besser zu planen.

Es wurde ein numerisches Model der HP³ Drucksondierung entwickelt um das Verhalten des granularen Materials während der dynamischen Sondierung zu untersuchen. Dieses Model besteht aus dem Instrumentenkörper welcher in ein granulares Material eindringt, sowie aus dem Schlagmechanismus welcher die Bewegung des Sensors erzeugt. Der Einfluss der Randbedingungen aufgrund der Einschränkungen im Labor sowie die Eindringperformance des Sensors in unterschiedlichen Böden wurden untersucht. Zudem ist der Einfluss des dynamischen Eindringens auf die Bodenbeschaffenheit ausgewertet.

Abstract

Numerical Modelling of Dynamic Cone Penetration into Martian Subsurface

The NASA InSight Mission is an unmanned space mission that will send a lander on Martian surface to investigate the interior of Mars. The expected launch will be in 2018 after the initial date of launch in 2016 had to be postponed due to technical problems. The major task of the mission is the study of the planets geological evolution by investigating the seismic activity and the planets heat flow. For this purpose, a seismometer (SEIS) will be placed on the surface and a heat flow probe (HP³) will be driven into the subsurface of Mars. The heat flow probe will penetrate 3 to 5 metres deep into the ground of Mars to avoid the influence of the solar radiation on the heat flow measurement and to allow measurement of the planetary temperature gradient using thermal sensors on the trailing cable at different depths.

HP³ contains an internal hammering mechanism that pushes the probe into the ground. The displacement of the probe during the penetration phase will be measured and used for the determination of Martian soil mechanical properties. Those information on the soil conditions and its soil mechanical properties provides conclusions on the planets history and shall be used to better plan future missions.

Therefore, a numerical model of the HP³ penetration progress has been developed to investigate the behaviour of granular materials during dynamic penetration. This model consists of a probe penetrating into granular material and a hammering mechanism that generates the movement of the probe. Investigations on the influence of boundaries in laboratory conditions and on the penetration performance of the probe in different soils are carried out. Furthermore, the influence of the dynamic penetration on the soil condition is evaluated.

Inhalt

1	Introduction	1
1.1	NASA InSight Mission.....	1
1.2	Heat flow and Physical Properties Probe (HP ³)	3
1.3	Landing site	5
1.4	State of knowledge on Martian soil properties.....	8
1.5	State of knowledge in pile installation	10
1.6	State of the art in numerical modelling	11
1.7	Laboratory conditions for HP ³ penetration tests	15
1.8	A brief introduction into the Material Point Method	17
2	The Discrete Element Method	20
2.1	The general DEM formulation	20
2.2	The Normal Contact Model.....	21
2.3	The Rolling Resistance Model	22
2.4	The Tangential Friction Model.....	30
2.5	The elastic-plastic yield criterion for frictional contacts.....	33
2.6	Validation of the Contact Models.....	36
2.7	Neighbor lists.....	38
2.8	The initial filling process.....	39
3	Calibration.....	41
3.1	Material.....	41
3.2	Angle of Repose	42
3.3	Triaxial Shear Test.....	44
3.4	Oedometer Test.....	45
3.5	Calibration Results	47

4	The Penetration Model.....	52
4.1	Simulation preparation	54
4.2	Validation of particle size scaling for the penetration simulation.....	55
4.3	Simulation of quasistatic cone penetration tests	62
4.4	Simulation of dynamic penetration of HP ³	65
4.4.1	Plane strain model with prescribed displacements.....	74
5	A Pile Drive Model implemented in Matlab	80
5.1	An analytical approach for the penetration resistance	81
5.2	Time Integration	84
5.3	Application	85
5.4	Simulation results	85
5.5	Friction fatigue and the influence on the HP ³ performance	89
6	Concluding remarks	92
7	Bibliography	95
8	Appendix.....	100
8.1	Simulation model structure	100
8.2	Modified rolling model sbjp.....	102
8.3	Modified rolling model stone2.....	114
8.4	Modified rolling model dahl2	131
8.5	Modified tangential model dahl	151
8.6	MatLab model for pile driving	164

Formelzeichen und Abkürzungen

Große Buchstaben

A	[m ²]	Cross-sectional area of penetrator
ALE	[]	Arbitrary Lagrangian Eulerian
B	[m]	Cone diameter
CPT	[]	Cone Penetration Test
CTX	[]	Context Camera
DEM	[]	Discrete Element Method
DLR	[]	Deutsches Zentrum für Luft- und Raumfahrt
E_{50}	[Pa]	Secant modulus at 50 % peak strength
$F_{Coulomb}$	[N]	Coulomb friction force
$F_{damping}$	[N]	Damping force
FEM	[]	Finite Element Method
F_n	[N]	Normal contact force
F_t	[N]	Tangential contact force
$F_{t,max}$	[N]	Max tangential contact force
G	[Pa]	Particle shear modulus
HiRISE	[]	High Resolution Imaging Science Experiment
HP ³	[]	Heat flow and physical properties probe
HRSC	[]	High Resolution Stereo Camera
I	[kgm ²]	Particle rotational inertia
IDA	[]	Instrument deployment arm
JPL	[]	NASA Jet Propulsion Laboratory
K_P	[]	Proportional constant for controller
K_D	[]	Differential constant for controller
MPM	[]	Material Point Method
$M_{r,plastic}$	[Nm]	Rolling resistive torque at mobilisation
NASA	[]	National Aeronautics and Space Administration
N_γ, N_q, N_c	[]	Bearing capacity factors
PFC ^{2D}	[]	Two-dimensional Particle Flow Code
R_{shaft}	[N]	Shaft resistance
R_{tip}	[N]	Tip resistance
SEIS	[]	Seismometer
SPH	[]	Smooth Particle Hydrodynamics
STATIL	[]	Static tilt sensor
Y	[Pa]	Particle Young's modulus

Kleine Buchstaben

d_{50}	[m]	particles' mean grain size
dt	[s]	time step size

dx	[m]	Relative displacements between two contacting particles
dx_e	[m]	Relative elastic displacements
dx_p	[m]	Relative plastic displacements
g	[m/s ²]	Gravitational constant
k_n	[N/m]	Particle normal stiffness
k_t	[N/m]	Particle tangential stiffness
$k_{t,prime}$	[N/m]	Particle tangential stiffness for primary loading
$k_{t,un/re}$	[N/m]	Particle tangential stiffness for un-/reloading
k_r	[N/m]	Particle rolling stiffness
m	[kg]	Particle mass
p_d	[m]	Distance between two particles' centre
r	[m]	Particle radius
r_v	[m]	Radius of particle v
r_w	[m]	Radius of particle w
r_{eff}	[m]	Particles' effective radius
s_γ, s_q, s_c	[]	Shape factors
t	[m]	Depth
v_t	[m/s]	Relative tangential velocity
\vec{v}	[m]	Position vector to a particle centre
\vec{w}	[m]	Position vector to a particle centre

Griechische Buchstaben

φ	[°]	internal friction angle
δ_n	[m]	overlap between two particles
μ_r	[]	Coefficient of rolling resistance
μ_t	[]	Tangential friction coefficient
μ_{inter}	[]	Interface friction coefficient
θ	[°]	Rotation angle of a particle
$\theta_{elastic}$	[°]	Elastic rotation of a particle
σ	[Pa]	Stress
σ_v	[Pa]	Vertical stress
σ_h	[Pa]	Horizontal stress
ε	[%]	Strain
γ_o	[N/m ³]	Soil specific weight above the tip
γ_u	[N/m ³]	Soil specific weight underneath the tip
ρ	[kg/m ³]	Soil density
ΔT_{crit}	[s]	Critical time increment
ω	[1/s]	Angular velocity

1 Introduction

1.1 NASA InSight Mission

The NASA InSight Mission will investigate the deep interior of Mars to determine the inner structure of the red planet more precisely and observe its geological processes. Since the inner structure of Mars has not changed as much as on Earth due to less geological activity, the mission will provide insights into the planets earliest evolution. The information about the geological evolution on Mars will reveal new knowledge on the evolution of all rocky planets, including Earth. So far, the knowledge on the inner structure of Mars is given by gravity and topography analysis as well as from magnetic data from Mars Global Surveyor (Williams 2008). The InSight Mission will now provide seismic and thermal data from the interior and thus will make it possible to create more accurate models of the planets structure and reveal the planets origin.

For this task, a lander will be send to Mars carrying the two key instruments: the SEISmometer (SEIS) and the Heat flow and Physical Properties Probe (HP³), see Figure 1. The interior structure of Mars will be determined much more precisely than in the past by measurements of the seismometer. Furthermore, the occurrence of tectonic activities on Mars as well as meteor impacts will be detected by SEIS. Although, SEIS will not be the first seismometer on a Mars mission, it will be the first seismometer placed directly on the Martian surface. Previous seismometers of the Viking Mission in 1976 were mounted on the lander deck, so that the motion of the lander structure itself manipulates the results. The InSight seismometer is mounted on a steerable mounting, so that the orientation can be adjusted after deployment. The system will be fixed by a conical tip that dips into the sandy surface to guarantee a stable connection between the sensors and the Martian surface (Christensen & Knapmeyer-Endrun 2016). An additional covering protects the seismometer against surface winds.

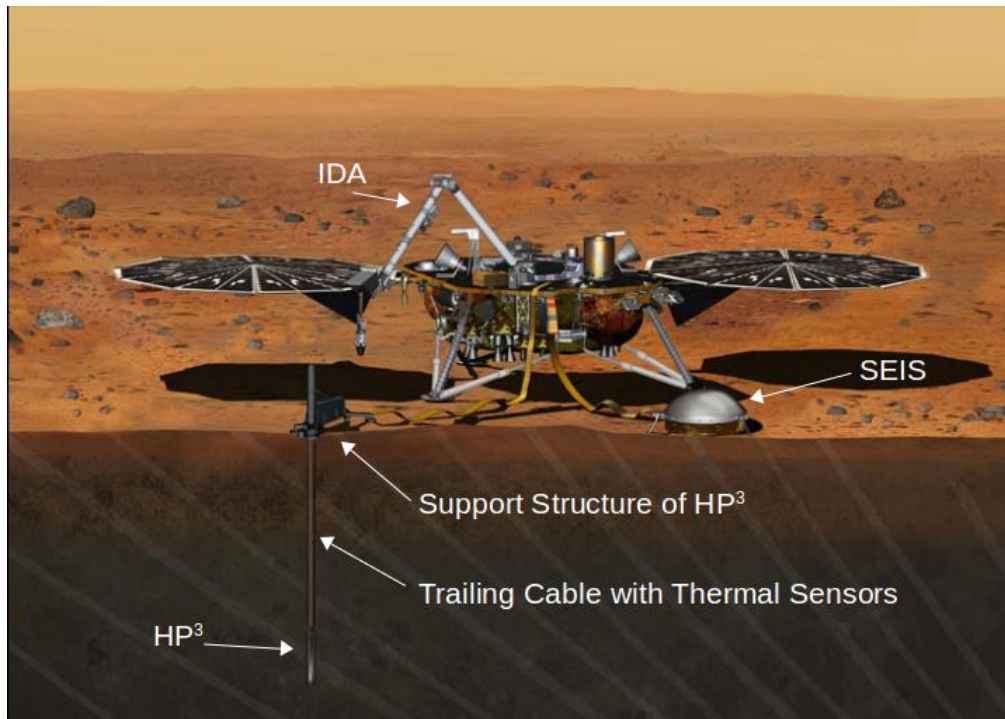


Figure 1: The InSight lander (NASA/JPL-Caltech 2015a).

The inner heat flow of the planet will be recorded by the thermal sensors of HP³. The HP³ instrument consists of a heat flow probe including a hammering mechanism to penetrate itself into the ground. A trailing cable for data exchange and power supply connects the probe with a support structure on the surface. The thermal sensors are mounted at the probe and every 10 cm on the trailing cable so that the temperature can be measured in different depth at the same time. Thus, a temperature gradient of the planet can be determined. The thermal sensors have to be installed in 3 to 5 metres depth to avoid the influence of the daily and annual variation of the solar radiation on the thermal measurements. The penetration progress will take about 30 Sols (Martian days) and around 10.000 hammer strokes approximately. The position of the probe will be recorded by measuring the extended length of the trailing cable and the orientation of the probe will be determined by a tilt meter. The penetration rate will provide information on the soil mechanical properties. Therefore, numerical models of the penetration process are developed to understand the mechanics of the soil during the penetration and to derive the soil mechanical parameters from back-calculations.

Further soil investigations will be done using the instrument deployment arm (IDA) mounted at the InSight lander. The major task of the IDA is the placing of SEIS and HP³ on the Martian surface. Besides, the IDA has a scoop that can be used to grade the surface or displace boulders if there is no space for the deployment of the instruments. Another suggestion for the usage of the IDA is the excavation of a trench and the creation of a stable sand pile by pouring out the excavated material. The shape of the trench and the angle of the sand pile will be analysed by a camera mounted at the IDA. This analysis will reveal the angle of

repose of the sand which corresponds to the critical state friction angle for a cohesionless sand. Thus, the usage of the IDA for soil tests will provide information on the soil strength parameter, i.e. the internal friction angle at critical state.

The mission was first planned for launch in March 2016 but had to be postponed for 2 years, caused by a technical problem on the seismometer. The reason was a leakage of the vacuum vessel that enshrouds the main sensors and is needed to enable a high sensitivity for measuring even the smallest ground movements in the high frequency range. After the deployment and installation phase of SEIS and HP³ is finished, the monitoring phase will take 1 Martian year which corresponds to 687 days on Earth. It will take another 7 month for the deliveries of all data and the mission will finish approximately 3 years after launch.

Table 1: Time schedule of InSight Mission

Task	Duration
Cruise	6,5 months
Instrument Deployment	60 sols or 58 days
Surface Monitoring	1 Martian year or 687 days
Final Data Deliveries	7 months

1.2 Heat flow and Physical Properties Probe (HP³)

The Heat flow and Physical Properties Probe is a sensor that penetrates itself 3 to 5 m into the Martian surface. The position of HP³ is provided during its penetration into the granular material and reveals information on the soil mechanical parameters at the landing site. After the penetration process, it measures the temperature of the planet over one Martian year. Therefore, the HP³ penetrator features thermal sensors on the penetrator as well as on the trailing tether to determine the temperature gradient. In addition, heater foils are installed on the penetrator for active thermal conductivity measurements. The information on the thermal conductivity and the temperature as well as the temperature gradient reveal the planet's heat flux.

The HP³ drive system consists of the hammering mechanism and a static tilt sensor (STATIL) which is fixed by shock isolation springs to protect it from the fast acceleration due to the hammering action. The hammering mechanism consists of a brake spring, a suppressor mass, a roller with a cylindrical cam, force springs and the hammer mass (Figure 2). The trailing cable (science tether) connecting the

penetrator and the structure on the surface is needed for the power supply and the data transmission. Besides, there are also thermal sensors on the trailing cable to measure the temperature in the ground in different depths. The penetrator's outer casing is a cylindrical tube with a diameter of 18 mm and an ogive tip. The length of the penetrator is 353 mm.

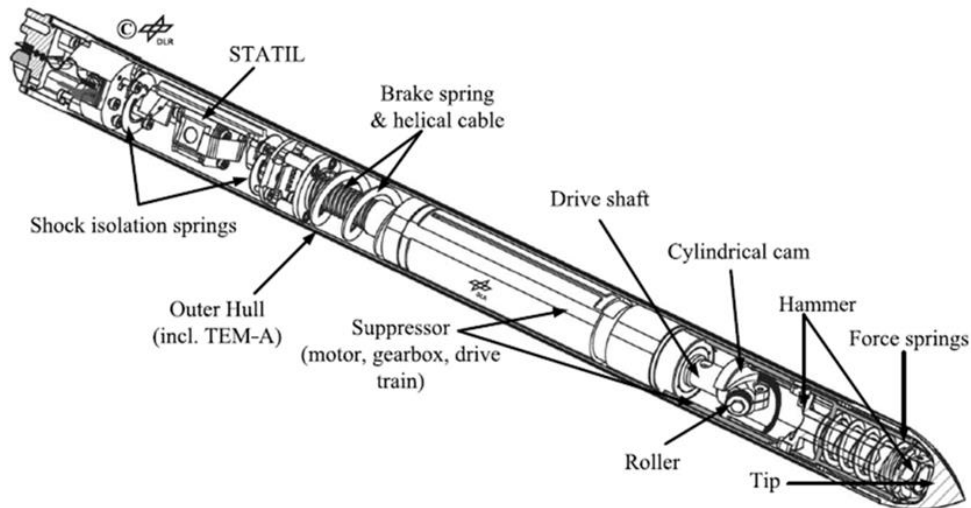


Figure 2: Interior of HP³ penetrator and its hammering mechanism, Lichtenheldt et al. (2014)

A hammering cycle begins with the loading of the force springs that connects the hammer and suppressor masses. For this purpose, the roller starts to rotate and pulls the hammer towards the suppressor to load the force springs. At the end of the loading phase there is a gap in the cam so that the roller loses the contact and the hammer is accelerated by the force springs. The hammer mass is pushed towards the tip, while the suppressor mass moves in the opposite direction. The hammer mass hits the tip while the suppressor mass is slowly decelerated by the brake spring which connects the suppressor with the rear casing. The mechanism drives the penetrator only if a sufficient shaft friction is present that prevents the penetrator from moving backwards due to the backward motion of the suppressor mass. After the motion of the suppressor is slowed down, the loaded brake spring accelerates the suppressor mass again and a second stroke due to the suppressor is generated. Further minor strokes of the oscillating system may occur with low impact energy. Such a loading cycle is repeated a few thousand times to drive the penetrator into a depth of 3 to 5 m below the surface. The driving mechanism is not gravity driven like usual vibratory penetrations, which makes it applicable also in low gravity environments.

1.3 Landing site

The landing site selection for InSight is done by the Jet Propulsion Laboratory (JPL) in Pasadena and is managed by Dr. Matthew Golombek. A detailed description of the landing site selection process considering the planning of the landing phase as well as the deployment and proper function of the science instruments is given by Golombek et al. (2016).

The InSight landing site is located in the western Elysium Planitia, an equatorial region of Mars. The local altitude is below -2.5 km and thus it is low enough for a sufficient atmospheric density to slow down the lander at descent. The estimated landing area is determined by ballistic entry and landing simulations considering uncertainties in the position and orientation of the lander at entry as well as deviations in the atmosphere and in the aerodynamics of the lander. The resulting landing area is an ellipse with a size of 130 km by 27 km that has to meet the requirements for landing safety and instrument deployment. For the landing safety, a smooth terrain with a radar reflective surface is needed to measure the altitude and velocity of the lander at descent precisely by the landing radar. Furthermore, a load bearing surface is required to bear the load of the spacecraft at touch down as well as for the proper function of the science instruments. Additionally, a 3-5 m layer of fragmented rock is needed for the penetration of the heat flow probe (HP³) to record the planets heat flow from the interior.

In the case of the InSight landing area, a smooth terrain is characterised by a small number of steep slopes and a low rock abundance. The slopes at the selected landing site are required to be less than 15° at a length scale of 1 to 5 m as well as for 84 m length scale. The slope angle below 15° on the small length scale is required to avoid a tip over of the lander at touchdown while the inclination of the terrain at a length scale of 84 m is required to determine the landers velocity precisely at descent by radar. Furthermore, the levelling system of SEIS can only compensate maximum slope angles of 15°. The slopes at the landing site are evaluated by digital elevation models that are based on images from the orbiters cameras HiRISE, HRSC and CTX. The HiRISE and CTX camera are mounted on the Mars Reconnaissance Orbiter of NASA whereas the High Resolution Stereo Camera (HRSC) is provided by the Mars Express orbiter of ESA. From these digital elevation model data, a map for slopes at 84 m length scale was derived and ensures that the slope angles at the landing region rarely exceed the restrictions of 15° (Figure 3). The data from the CTX camera reveal that the area with slopes exceeding 15° covers about 0.66 % of the landing area, which is below the requirements of 1 %.

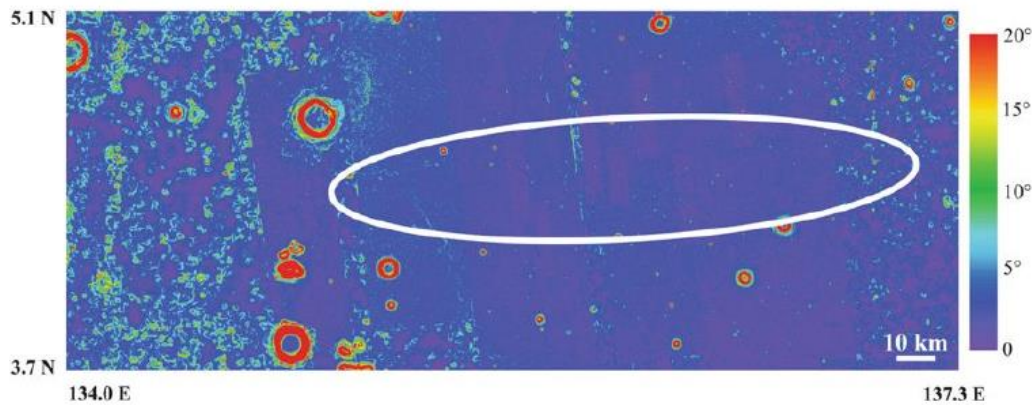


Figure 3: Slope map from digital elevation model images created by HRSC, CTX and HiRISE images of the landing ellipse E9 for slopes at 84 m length scale (Golombek et al. 2016).

Furthermore, maps for slopes at 2 m length scale were developed and reveal that only about 0.1 % of the surface at the E9 landing site exceed the restrictions of 15° at this length scale. With respect to the slopes at the landing site, the requirements of a smooth terrain are fulfilled.

Besides the slopes, there is the hazard of surface rocks that could damage the lander at touch down or impede the deployment of the seismometer and the heat flow probe. A terrain map of the landing site from open to close of the launch period is shown in Figure 4, where the dominant area in green represents a smooth terrain while the smaller light purple area in the north east of the map is a more ridged terrain. A rock abundance of about ~1.2 % is present at the landing site, which is far below the requirements of 10 % and improves the landing safety as well as the possibility for the deployment of the instruments. The rock sizes are determined by an automated analysis of their shadows, where rocks smaller than 45 cm are not hazardous for the lander and rocks up to 3 cm can be ignored for the deployment of the instruments. The instruments can be deployed by the arm of the lander within an annular workspace in a distance of 0.5 to 2 m from the lander over an arc of 180°. Both instruments need to be placed on a load bearing surface with low dust deposits. The HP³ penetrator needs a layer of regolith (i.e. fragmented rock) to a depth of at least 3 m to be able to penetrate into the ground and avoid the influence of solar radiation on the thermal measurements. Observations of local craters reveal that a layer of fine-grained regolith overlaying a rocky layer is present and has a thickness of ~10 m depth. This is the result of investigations on the presence of rocky ejecta for different crater sizes.

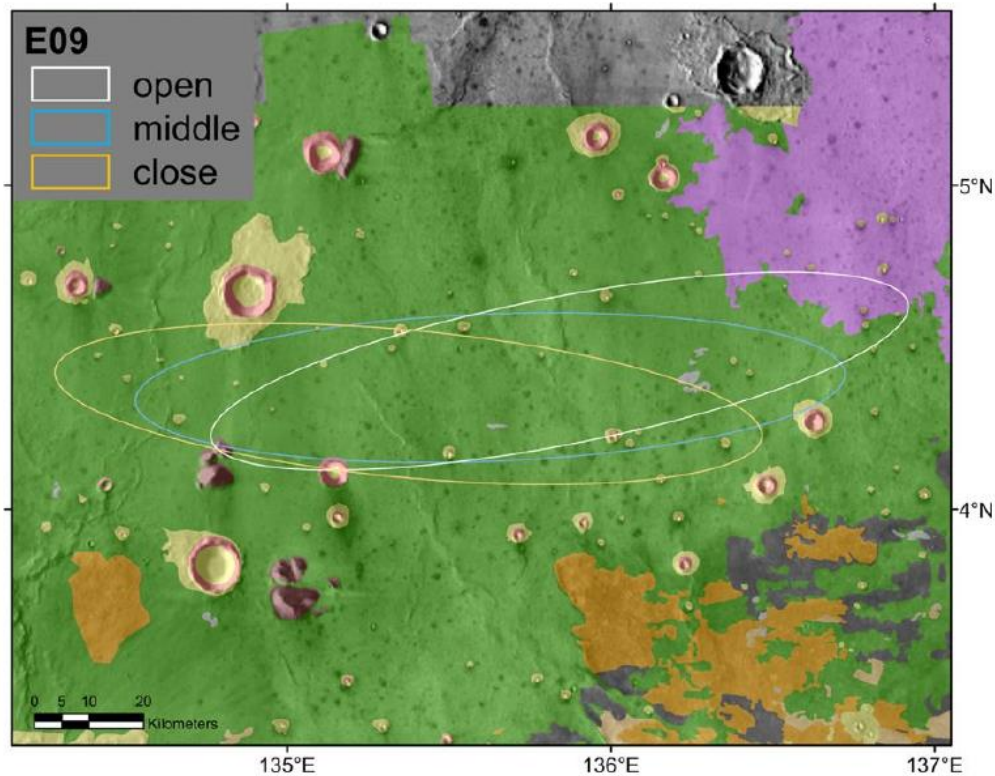


Figure 4: Terrain map of the final landing ellipse E09 for three different launch times, from open to close of the 2016 launch period (Golombek et al. 2016).

The digital elevation model reveals in total a smooth terrain with less than 0.5 % area that exhibit slopes larger than 15° at 1-5 m length scale and is thus smoother than previous landing sites of the Opportunity rover and the Phoenix lander mission. Thermal image data provided by the orbiters suggests that the landing site is covered by cohesionless fine sand. The slight seasonal variations of the thermal inertia let assume a constant layer of at least 0.5 to 1 m below the surface.

Due to the global location in the equatorial region, there is no liquid or frozen water expected within 5 m below the surface. Investigations of high-resolution images of steep slopes indicate that the terrain is shaped by eolian processes without any water or ice related characteristics. The terrain is also shaped by craters that are determined to be mostly secondary craters from an impact 700 km to the north east of the InSight landing site. The main crater is called Corinto and its date is estimated to be prior to 0.1 Million years ago. There are several secondary craters in the landing region with depth/diameter ratios of about 0.05 which is lower than expected and the interior slopes are rarely at the limit of 15° . The secondary craters cover 1.5 % of the landing region, where their contribution to the average slope distribution is small.

Altogether, there is a good chance of a successful mission owing to the detailed selection of a safe landing region.

1.4 State of knowledge on Martian soil properties

In the past, several Mars missions have already studied the soil properties at different landing sites to get a better knowledge of the geological evolution on Mars and to better plan future missions. A topographic map with a colour coding for the elevation is shown in Figure 5.

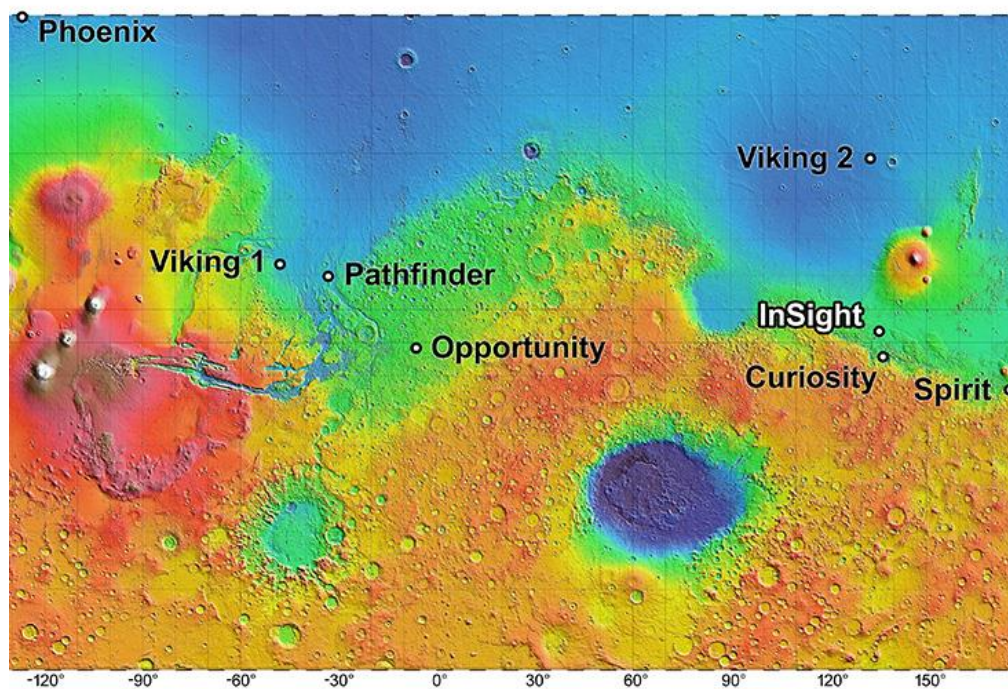


Figure 5: Landing sites of Mars missions in the past and the planned landing site for InSight. (NASA/JPL-Caltech 2015b)

The Viking landers in 1975 were the first missions on Mars performing soil tests for the determination of soil mechanical properties. For this purpose a robotic arm with a scoop was used to dig into the Martian surface investigating the stable slope angles of poured materials and performing bearing tests. The material at the Viking landing site is assumed to represent the material at most areas of the equatorial region (Moore 1989). The drift material at the landing site is a scoured material on the surface consisting of very fine grains in μm size and behaves very soft, which can be hazardous for rovers traversing. This loose material reveals a low bulk density of 1000 to 1300 kg/m^3 . The internal friction angle of the drift material is between 16 and 20 degrees and the cohesion ranges from 0 to 3.7 kPa caused probably by cementation. In comparison, the underlying crusty to cloddy materials exhibit internal friction angles of 30 to 39 degree which is similar to values of terrestrial sand and indicating a material with higher bearing capacity. The cohesion of this materials is below 3.2 kPa and the bulk density is estimated to be $1400 \pm 200 \text{ kg}/\text{m}^3$. Further clumped blocky materials with cm sized clumps are present having an internal friction angle of 30.8 ± 2.4 degree and a bulk density of $1600 \pm 400 \text{ kg}/\text{m}^3$. The cohesion of the blocky materials is in the range of 2.2 to

10.6 kPa. Moreover a few large rocks are present at Viking landing site but there will be even less at the InSight landing site.

The Mars Pathfinder mission in 1996 performed soil mechanical tests using the Imager for Mars Pathfinder (IMP), i.e. a camera on the lander, a second camera on the rover and the wheels of the rover. The internal friction angle of the soil was determined by information on the angle of repose from images of excavated tailings from wheel digging and from the measured motor currents during the wheel trenching. The values of the internal friction angle varied depending on the layer and the site. Lower friction angles of 28 and 35 degrees depending on site were observed by Moore (1999) in a thin layer at the first few cm of digging, while below that layer higher friction angles of 37 and 41 degrees were observed. The low friction angle of 28 degrees is attributed to drift material that overlays a cloddy deposit. The angles of repose measured from IMP images were between 32 and 38 degree. The cohesion were measured to be quite low with values below 1 kPa.

The Mars Exploration Rovers (MER) Spirit and Opportunity in 2003 used their wheels to trench and scuff the surface materials on Mars for determination of soil strength parameters. For this purpose the mid and rear wheels were locked while the front wheel digs into the soil. The data provided by the rovers were the motor currents, the load on the wheels and the depth of the buried wheel from images. For the determination of the internal friction angle, the material was first trenched to create a tailing pile where the cohesive bonds are assumed to be broken after trenching. Then shear tests were carried out using one wheel of the rover digging into the excavated material. The obtained friction angles of the material by Sullivan (2011) turn out to be between 30 and 37 degree which corresponds with dry sandy soils on earth. The cohesion of the soil was determined by digging tests on undisturbed material and the knowledge of the internal friction angle from the tests on tailings. The values of cohesion ranged depending on site from 0 to 2 kPa and from 0 to 11 kPa, but in the case of high cohesion the uncertainties were up to +/- 3.9 kPa. A low cohesion of 2 kPa can also be obtained without any bonding from dry sharp edged sand or if contents of silt and clay are present.

The Phoenix Mars Mission in 2008 used optical and atomic force microscopy to determine the shape and the grain size of Martian dust. The shape of the particles and their grain size distribution provide information about the particles' transport mechanisms and their weathering processes, Pike et al. (2011). The atomic force microscope was used to investigate particles of micrometre size, whereas the optical microscope was used for particles of millimetre size. Furthermore, a mass spectrometer was used to determine the chemical analysis of the collected materials. The landing site of Phoenix is in the Martian northern hemisphere at the polar region, where frozen soil is present near the surface influencing the mechanical response of the soil. Shaw et al. (2009) determined the soil strength parameters, internal friction angle and cohesion, of the Martian soil at the Phoenix landing site. For this purpose, the robotic arm on the Phoenix lander was used to

excavate trenches and create dump piles. The internal friction angle was determined by measuring the angle of the dump pile slopes, assuming that all cohesive bonds were broken during the excavation of the material. The cohesion of the soil was calculated from the resisting force during the excavation, which is determined from the motor currents and the arm kinematics. There are uncertainties in the calculation of the force and the position of the scoop due to the stiffness of the robotic arm, so that the obtained values for cohesion are not very reliable. The determined values for cohesion are below 2 kPa, where the largest cohesion is attributed to ice in the soil. The angle of internal friction is assumed to be between 33 and 42 degrees based on the corresponding dump pile slope angles.

Observations of landslides on Mars were done by Perko et al. (2006) and provide information on the soil strength parameters. For this purpose, high resolution images and laser altimeter measurements of Mars Global Surveyor orbiter were used for a stability analysis of natural slopes in different regions. In the Hematite area the steepest angle of natural slopes turn out to be about 30 degrees, which corresponds to sandy soil on earth. The natural slopes at Gusev Crater reveal a steepest angle of 38 degrees, where the soil is characterised as a mix of coarse and fine materials. The material at this area is assumed to be densely packed because of its high thermal inertia of $450 \text{ J/m}^2\text{Ks}^{1/2}$.

The knowledge of the soil mechanical properties on Mars helps to understand the geological processes that forms the shape of the planet and provides guidance for future missions. The mean values of internal friction and cohesion of Martian soil lead to the conclusion that the materials are similar to terrestrial sandy soils. The differences of the internal friction angle depending on the local site indicate that there are different origins and mechanisms that shaped the Martian surface.

1.5 State of knowledge in pile installation

The standard procedure for pile installation is the impact driven pile. For this purpose, a heavy weight is raised above the pile and released to use the impact energy to drive the pile into the ground. Another procedure is the vibratory installation of piles, where a vibratory hammer is installed onto the pile and transmits vertical vibrations into the pile. The vibratory hammer consists of rotating eccentric weights that are arranged in a way that the horizontal acting forces counterweight each other. The development of more advanced vibratory hammers makes it nowadays possible to drive even large offshore monopiles into the ground. The vibratory driven piles have the advantage of quicker installation times and reduced costs. Therefore, the vibratory installation procedure is of interest for many applications, whereby the impact on the surrounding soil and the pile capacity is still not fully understood. Galavi et al. (2017) investigates the vibratory installation process of offshore monopiles using the Material Point Method to understand the influence of the installation type on the bearing capacity.

Even though the installation procedure of HP³ is different to a vibratory installation there are some similarities. Both installation methods have an alternating motion in short periods. The temporal evolution of a penetration resistance is similar for small vibration amplitudes, see Vogelsang et al. (2017). The cyclic shearing of the soil at the shaft is present in both cases and changes the soil structure in similar ways. The difference is that the vibratory hammers are driven by gravitational loading, whereas the HP³ mechanism needs solely a sufficient shaft friction to absorb the backward accelerations.

The current calculation methods on pile installations are based on impact driven piles, where effects of vibratory installation are not considered. Therefore, numerical methods are required to compute the penetration rate of HP³.

A research project by the Deep Foundations Institute (2015) investigated the influence of the installation procedure on the axial and lateral bearing capacity of driven piles. It was found that the axial capacities of vibrated piles was always less than for impact driven piles. The average capacity was reduced by 20 % when using the vibratory installation procedure. Whereas, the axial capacity can be increased by a followed impact installation to the full depth. The lateral capacity was found to be less influenced by the installation procedure, although it exists less data of experiments to confirm this.

The vibratory pile installation is a promising technique with short installation times, although the influence on the surrounding soil is not completely known yet. Therefore, it is necessary to investigate this penetration process in detail with the help of numerical models. There is an enormous amount of research on this topic that provides new insights, e.g. Grabe et al. (2013) investigated already the soil changes from a deep vibration compaction with the help of a coupled Eulerian-Lagrangian method.

1.6 State of the art in numerical modelling

Numerical simulations of dynamic cone penetration tests are quite rarely in literature whereas simulations of quasistatic cone penetration tests are more common. Different numerical methods have been used to simulate driven piles with constant velocity. The common methods that are currently used for numerical investigation on cone penetration or pile driving are:

- Enhanced Finite Element Methods (FEM):
 - Material Point Method (MPM)
 - Arbitrary Lagrangian Eulerian (ALE)
 - Other similar FEM based methods

- Smooth Particle Hydrodynamics (SPH)
- Discrete Element Method (DEM)

The ALE and MPM approach apply optimisation of the elements' shape or particle-in-cell techniques to solve large deformations with the Finite Element Method. All FEM based solutions use a continuum approach for the representation of the simulated material. For this purpose, the discontinuous granular material is simplified as a continua, wherefore constitutive models are necessary to represent the stiffness and strength behaviour. A common model for the strength of granular materials is the Mohr-Coulomb failure criterion. Many constitutive models are based on this theory and extend it. Still, the dilative behaviour of soils and the associated softening is difficult to model with FEM based approaches. Besides this, the frictional contact between soil and penetrator is challenging to be modelled in finite element based methods and has a great impact on the penetration resistance. In the doctoral dissertation of Issam (2013) a MPM formulation was used to solve the large deformations in a deep penetration process. The installation process was modelled by a varying driving force to represent an impact driven pile installation. The penetration rate for one hammer stroke for different skin friction is shown in Figure 6. For a shallow penetration the rate per stroke varied between 12 cm, 6 cm and 4.5 cm depending on the skin friction of 0.0, 0.5 and 1.0, respectively.

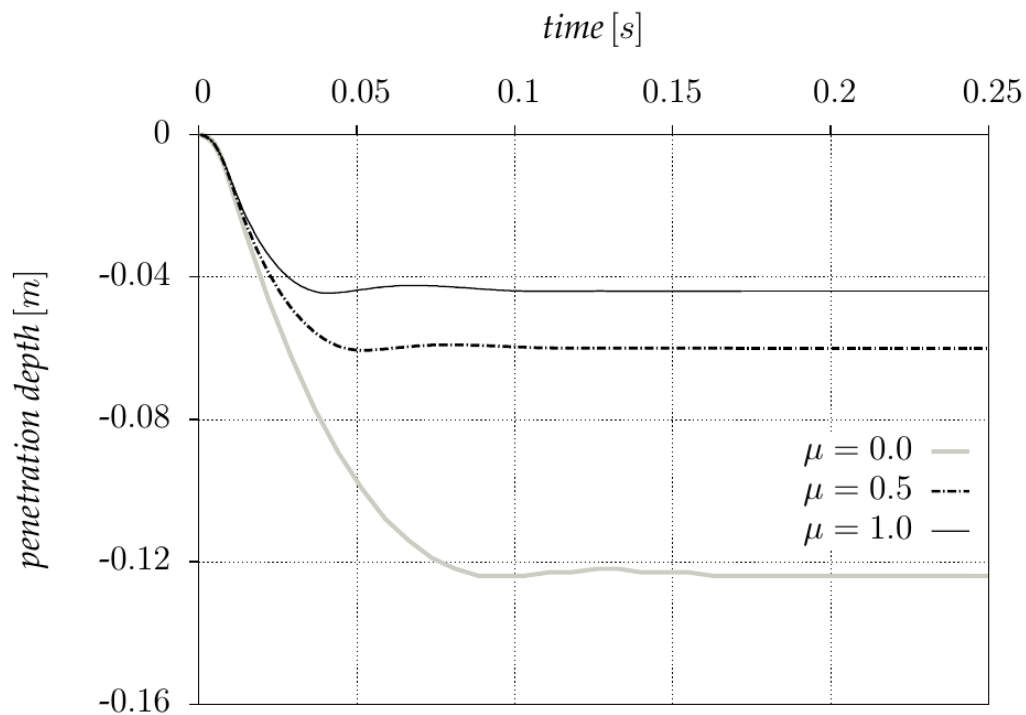


Figure 6: Simulation results of an impact driven pile using a MPM formulation. (Issam 2013)

Further simulations of a CPT using the Material Point Method were done by Ceccato et al. (2016). The cone penetration test under different types of drainage was investigated using a two-phase formulation for the MPM. In the two-phase MPM, the material points contain the information of the soil and the water. Besides fully drained and undrained behaviour, also a partially drained material can be simulated by this approach. The computational cost of MPM simulations of CPTs are in the same range as similar DEM simulations.

Simulations of cone penetration tests in sand using the ALE technique were done by Susila et al. (2003). The use of an auto-adaptive remeshing avoided mesh distortions and enables the simulation of a penetration up to 11 cone diameters. The penetration resistance for different internal friction angles and for different initial vertical stress was evaluated.

The SPH discretises a continuous field into a series of particles to solve partial differential equations. The material is subdivided into many particles (elements) where each particle is carrying physical quantities of the current state. The discretisation transforms the partial differential equation into an ordinary differential equation that can be solved by many integration schemes (e.g. Euler method). The material behaviour is governed by the differential equation. Thus, it is necessary to define constitutive models to describe the behaviour of granular materials like in the Finite Element Method. The advantage of SPH is that the material can be highly distorted because the particles can move freely and their adjacent particles are always updated. Kulak & Bojanowski (2011) applied the SPH in combination with the ALE for the simulation of a cone penetration test. SPH particles were used in the region near the penetrator where large distortions appear. The penetrator size has similar dimensions as HP³ and the resulting resistant force is comparable.

The DEM uses many individual elements representing a particulate material. The simplest and widely used elements are spherical particles, since it is not necessary to compute their orientation. The most common method in geotechnics is the soft-particle approach, where the particles can overlap among each other and contact forces are calculated and applied to the contacting particles. Further contact models, e.g. tangential friction, can be applied additionally to consider all necessary physics of the particulate material. The application of DEM in geotechnics is limited on small scale simulations due to the required amount of particles. The bulk behaviour of the particulate material results out of the contacts and particle movements. This allows to investigate particle scale phenomena without any presumptions made on constitutive models. Therefore, the particles' movements can be traced and changes in the soil structure can be investigated. Due to the discontinuous approach there are less restrictions in the way of modelling, e.g. interpenetration of different materials or generation of cavities in the material can be simulated. The DEM is less common used in geotechnical applications up to now, because its applicability on large scale simulations was limited by the

amount of particles and the associated computation time. Currently, due to new computer technologies and using parallel processing, it becomes more and more popular also for geotechnical applications where millions of particles have to be simulated.

Holmen et al. (2017) simulated a penetration test in a cylindrical tube, where the influence of different tip shapes on the penetration resistance is investigated and compared to experimental results. Since the tube size is limited, it was possible to simulate the real grain size by using 3.2 million particles with a mean particle diameter of 1.09 mm. The penetration was done quite fast with a penetration velocity of 2.5 m/s and 5 m/s. The simulations ran on a graphics processing unit which accelerates the computation. Furthermore, the simulation only considers translational motions while the rotational degrees of freedom are locked. This affects the behaviour of the granular material, but the results were still in good agreement with the experiments. Tran et al. (2016) reveal that the penetration resistance in constant velocity condition is only stable for penetration rates lower than 1.25 m/s. Consequently, the penetration resistance in the simulations of Holmen et al. (2017) is affected by inertia forces of the particles.

Tran et al. (2016) investigated the tip resistance of a constant and impact driven penetrometer with Itasca's software PFC^{2D}. It was observed that the impact driven probes involve always an elastic rebound after penetration, whereas for small penetration velocities the rebound is so large compared to the penetration that the penetrator is lifted back into its initial position. As a result, a minimum velocity of 0.5 m/s was determined to be necessary to penetrate into the granular material. The material was very dense packed with a porosity of 0.15 and the interparticle friction parameter is 1.0. This results in a very high resistance of about 2 MPa, which differs considerably from the observations presented in this thesis for the penetration of HP³.

Further simulations of cone penetration tests in DEM were done by Butlanska et al. (2014), who studied the influence of different boundary conditions and also the difference between free and locked rotation of particles. The effect of particle shape on the penetration resistance was investigated by Falagush et al. (2015) using different clumps of particles or by prohibiting particles rotation. Simulations of quasistatic penetration for the application on an earlier Mars mission were done by Zöhrer (2006), who investigated the penetration resistance for different tip angles and soil conditions. The use of DEM for penetration simulation is gaining more and more interest, since the computational time gets reduced with new technologies and more efficient particle codes.

After a first evaluation of the numerical methods that are used for cone penetration tests in literature, the MPM and the DEM approach were investigated in more detail.

1.7 Laboratory conditions for HP³ penetration tests

The penetration tests of HP³ are performed in a laboratory of the DLR in Germany. The performance tests for the deep penetration up to 5 m are done in an 80 cm wide cylindrical chamber with a depth of 5 m (Spohn, 2013). The container diameter should be at least 30 to 40 times the cone diameter (see Figure 7) which is a result of centrifuge cone penetration tests in dry sand by Bolton et al. (1999). This means that the cone diameter should be less than 2.6 cm to avoid influences due to the boundaries. Whether the same prediction can be made for the dynamic cone penetration is investigated by the numerical analysis in the following chapters. Therefore, the particles displacements and the induced stresses are compared for constant driven and dynamic driven probes. Besides the 5 m deep container, there is another 3 m deep and 60 cm wide cylindrical container for penetration tests at DLR (see Figure 8). The older 3 m deep test bed was used for the first penetration tests of HP³, whereas later on the 5 m deep test bed was constructed for the HP³ tests and primarily used. The results of the HP³ performance tests are discussed in chapter 4.

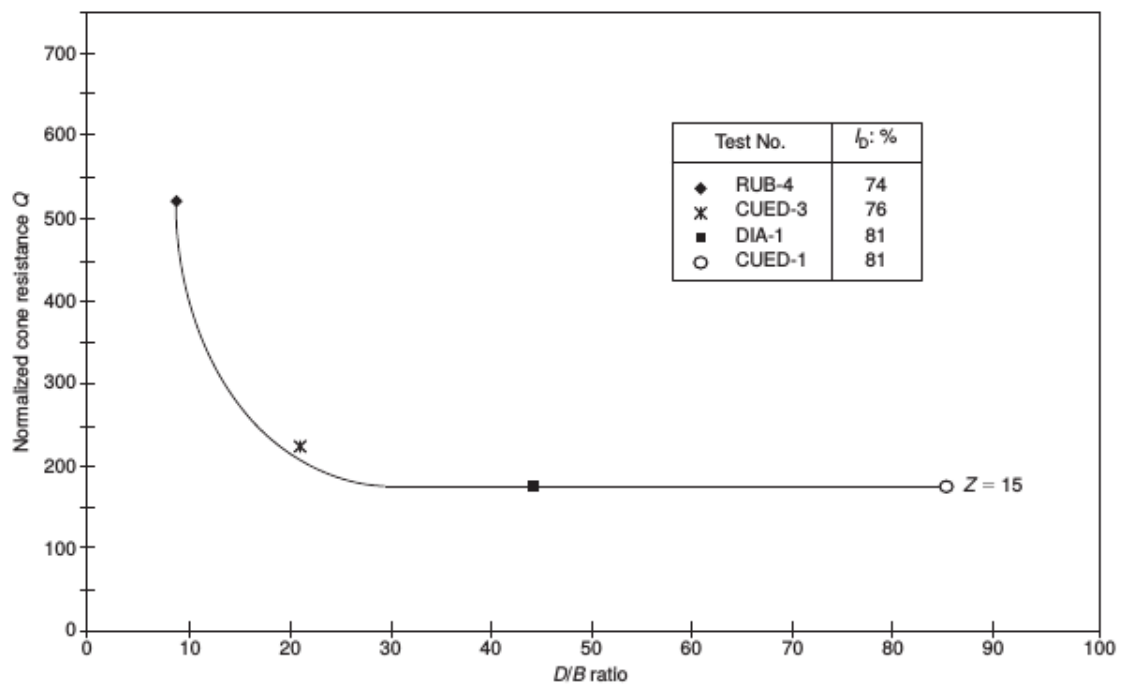


Figure 7: Effect of the container diameter to cone diameter D/B ratio on the penetration resistance by Bolton et al. (1999).

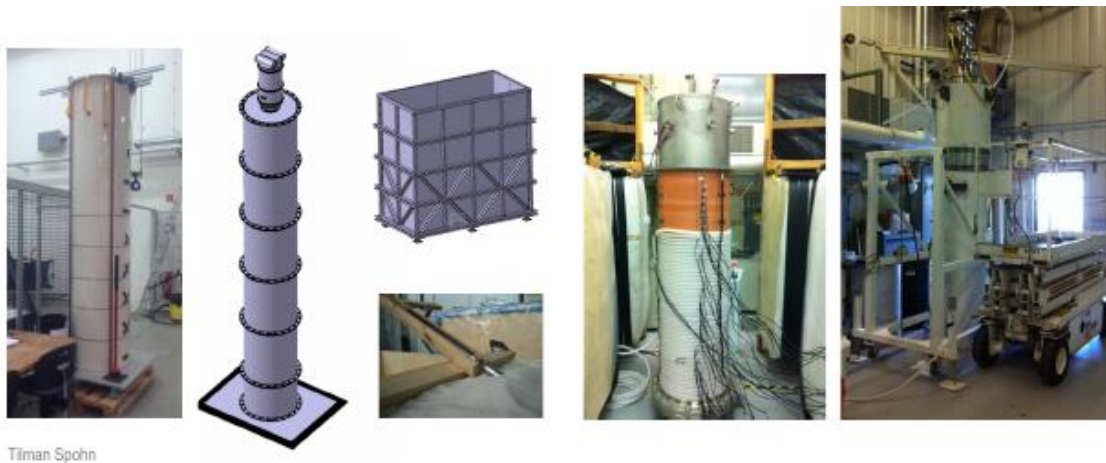


Figure 8: Deep penetration test bed with 3 m and 5 m height as well as the incline, geothermal and mechanical test bed of DLR and JPL (Spohn 2013).

For the tests at DLR, the atmosphere and gravitation is not adjusted, so that the earth environment is present. Whereas further tests in a Mars similar environment are performed at JPL in Pasadena, where a CO₂ atmosphere at 6 mbar and low temperatures is created for a 3 m deep test bed. Experiments in lower gravity are not really feasible because of the size and the duration of the test. However, numerical simulations provide the possibility of penetration tests in different gravitational environments.

The initial stress level, which is different from Mars to Earth due to the gravitational constants, has a significant effect on the penetration resistance. The centrifuge tests by Bolton et al. (1999) showed that the penetration resistance normalized with respect to the overburden pressure increases with smaller initial stress levels, see Figure 9. The plot shows the normalized resistance over normalized depth at different acceleration ratios N . The gravitation in the centrifuge was increased to 40g, 70g and 125g. A dense packed sample was used with a relative density of 0.96. Even though the normalized resistance is reduced by an increased stress level, the total resistance increases with higher initial stress levels.

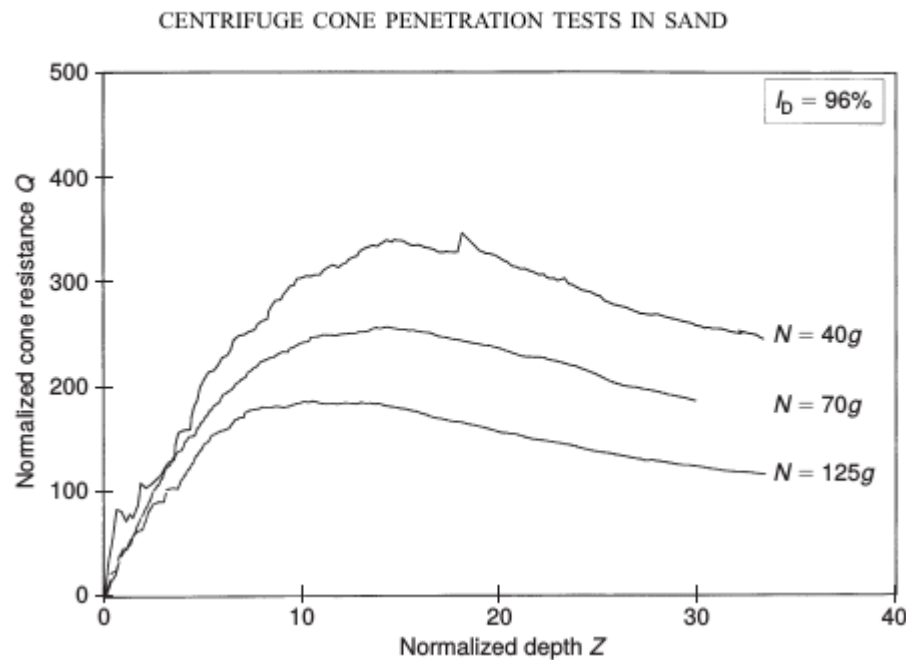


Figure 9: Effect of stress level on the penetration resistance by Bolton et al. (1999).

1.8 A brief introduction into the Material Point Method

The Material Point Method (MPM) was originally developed by Sulsky et al. (1993) and was first known as the Particle in Cell (PIC) method. Later on Sulsky & Schreyer (1996) called it the Material Point Method. The PIC method was already used earlier for simulation in fluid dynamics and was then adapted by Sulsky et al. (1993) for the application in solid mechanics. A first application of the MPM in geotechnical engineering was the simulation of a silo discharge by Wieckowski et al. (1999). The MPM is a finite element based method that is extended for the simulation of large deformations. Material points are introduced inside the finite element mesh to carry the information of deformation, stresses and other material properties. At each time step, the properties from the material points are mapped onto the mesh nodes, where the differential equation of the virtual work is solved. The strains and stresses due to the deformed mesh are saved again in the material points before the mesh is redefined or reset to its initial configuration. In this way, the material points can move behind the mesh, while the mesh is always updated at each time step. A graphical representation of this procedure is shown in Figure 10. The time domain is integrated by a semi-implicit scheme. Thus, the computational costs are reduced due to the possibility of larger time steps in comparison to an explicit time integration scheme. However, for high dynamic simulations small time steps are still required.

The grid-crossing of a material point can cause an unbalance force, where less particles per element increase the effect. A higher number of particles may solve this problem but increases the computational costs significantly. The error of grid-crossing can be reduced by different methods, see Issam (2013).

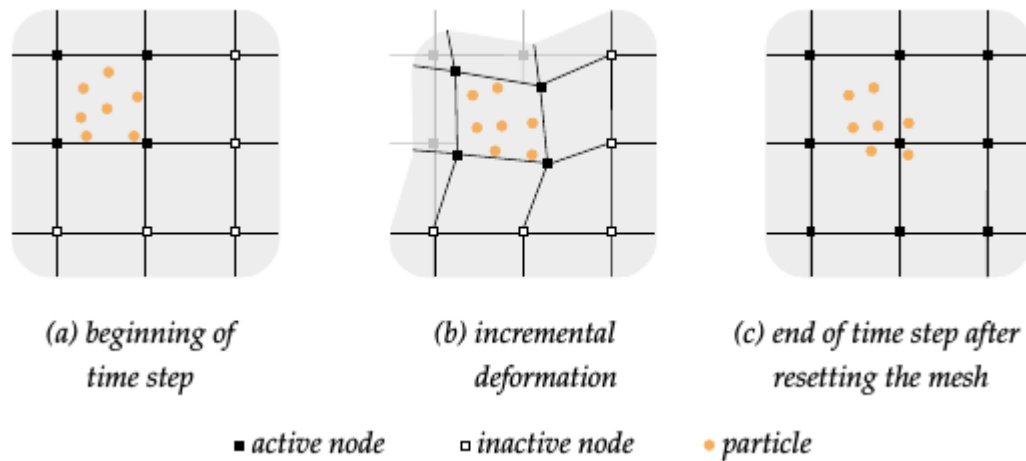


Figure 10: The basic concept of the MPM formulation. (Issam 2013).

The initial filling of the material points is important for accurate results and to prevent empty elements within a material. There are no empty elements allowed inside the material to solve the system. Therefore, virtual particles are introduced to avoid empty elements. These particles fill the empty elements with a small mass during the computation. The total mass of virtual particles should not be too large to obtain reliable results.

Even though material points are used to represent the granular material, the material behaviour is governed by the deformation of the continuum elements at each time step. Thus, it is necessary to apply constitutive models to describe the granular behaviour. The frictional behaviour in contacts needs also be defined by an appropriate algorithm to allow for the relative motion between two contacting bodies. In usual FE codes, the interface between two contacting bodies needs to be predefined, whereas in the MPM a separating of bodies and a colliding of bodies should be possible. For this purpose, an automatic detection of the contact surface is needed. An appropriate contact algorithm by Bardenhagen et al. (2000) was implemented by Issam (2013), but also current research on the detection of new contacts between two different materials in MPM is done by Hamad et al. (2017).

The MPM code by Deltares is currently more and more extended and improved in collaboration by the Anura3D MPM Research Community. The extension of a two phase formulation allows also for the simulation of water within the soil and the further research on the shortcomings improved the code a lot in the last years. The material point method is a practicable solution for large deformations but at the time the computational costs for a high dynamic simulation seemed to be immense.

The high processing times for the simulation of a quasistatic penetration test, where large time steps can be used and mass scaling was applied, suggested inappropriate large computation times for high dynamic simulations. Furthermore, the missing access to the code would have made it impossible to implement changes if necessary. For these reasons, the discrete element method was chosen for the simulation of the HP³ penetration.

2 The Discrete Element Method

The Discrete Element Method (DEM) was chosen for the simulation of the penetration process of HP³ since the modelling of large strains in dilatant materials is difficult to realise using a continuum-based approach (Butlanska et al. 2014). An advantage using the DEM that a constitutive model for the soil behaviour is not required. This allows to investigate the soil response without any assumptions on the soil behaviour beforehand. The non-linear stiffness and the complex strength behaviour of granular materials is automatically achieved due to the rearrangement and interaction of the discrete particles. However, assumptions on the interparticle behaviour have to be made instead. These will be discussed below. The DEM simulation of soil requires always a three dimensional model to capture the physics of the granular material. A two dimensional simulation of particles in a plane would model the behaviour of cylindrical rods instead. For the purpose of a plane strain simulation in DEM, it is still necessary to use a three dimensional model to reproduce the granular behaviour. However, it is possible to use periodic boundaries to reduce the simulation domain. The periodic boundaries allows particles to leave the domain at the border, while they re-enter on the other side of the domain. In this way, simulations using the assumption of an infinite half-space can be modelled in a small-scale test.

The use of discrete particles instead of continuum elements, such as in the finite element method, allows for simulations involving large deformations, separation of material and for interpenetration of different materials. The investigated materials are dry cohesionless sands that can be well modelled in the DEM since it needs only a few physical contact models. All of these contact models will be explained in this chapter.

The used DEM software is LIGGGHTS from DCS Computing developed by Kloss et al. (2012), which is an open-source software based on the LAMMPS code from Sandia National Laboratories.

2.1 The general DEM formulation

The functional principle of the DEM is the computation of the motion of a large number of particles. The material is modelled by many particles, where each particle stores a position in x-, y- and z-direction and a radius. The distance p_d of each particle to a wall or another particle is calculated at each time step:

$$p_d = |\vec{v} - \vec{w}|, \quad (1)$$

where \vec{v} and \vec{w} are vectors that define the positions of two different particles. If the distance p_d is smaller than the sum of the corresponding particles radii, an overlap δ_n is calculated as

$$\delta_n = r_v + r_w - p_d \quad (2)$$

with the corresponding particles radii r_v and r_w . A separating force F_n depending on the overlap is applied to both particles, see Figure 11. The position of the particles is updated for each time step by an explicit time integration scheme. The LIGGGHTS code uses a velocity Verlet integrator with half-step velocity to solve the time integration (Verlet 1967).

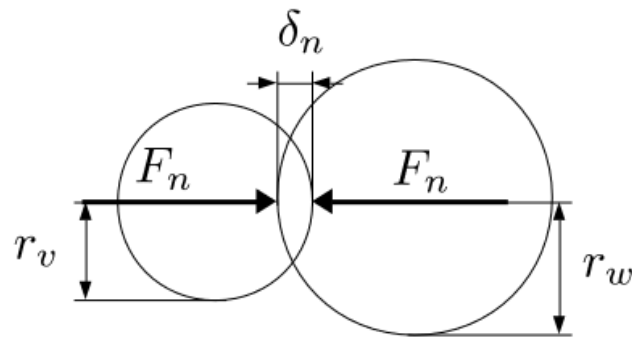


Figure 11: A sketch of the soft particle DEM.

In addition to this, different contact models are used to add further contact forces and torques to overlapping particles that can be e.g. sliding friction or rolling resistance. The contact force F_n between two particles can be either calculated linearly dependent on the overlap by using the Hooke's law or considering also the contact area by the Hertzian contact theory. In the following chapters, the contact models that are used for the simulation of a dry cohesionless sand will be explained in more detail.

2.2 The Normal Contact Model

The LIGGGHTS code provides two different normal contact models to determine the normal force between the particles based on the Hertzian and the Hookean contact mechanics. The simpler model is the Hookean normal model that calculates a normal contact force F_n linearly dependent on the overlap δ_n

$$F_n = k_n \cdot \delta_n \quad (3)$$

with the particles stiffness k_n being a constant value. This approach does not take into account the change of the contact area with overlap and therefore it can result in unphysical large overlaps of particles.

The Hertzian contact theory is well known for solving the contact mechanics of point or line contacts, where an infinite stress occurs theoretically caused by a zero contact area. The Hertzian contact model in LIGGGHTS considers the change of the contact area of two spherical particles that overlap. For this purpose, the stiffness of the overlapping particles is not constant as it is for the Hookean contact model, but it is dependent on the square root of the overlap. This causes a progressive behaviour of the contact force with overlap and avoids large overlaps of the particles (Figure 12).

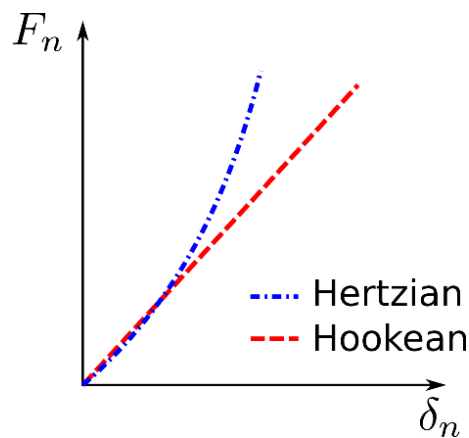


Figure 12: The characteristic curves of normal force with overlap.

Both contact models are fully elastic, which means that all the deformation are recovered at unloading.

Besides the elastic normal contact force, a damping force is applied to the normal contact model that defines the amount of rebound after a collision. The coefficient of restitution is used as an input parameter to specify the damping ratio in the normal contact. It defines the ratio of relative velocities of two particles after and before collision. Due to the damping force, a temporary tension force between two separating particles is possible. In order to avoid those tension forces, the damping force is limited so that the normal contact force is always a repulsive force. Therefore, the normal contact model in LIGGGHTS has to be extended by the keyword *limitForce on*.

2.3 The Rolling Resistance Model

A useful simplification in the DEM to be able to simulate a large amount of particles is the spherical shape of the particles. Due to the spherical form of the particles, the inertia tensor becomes a constant value owing to the point-symmetry. Thus, it is not necessary to compute the orientation of the particles. This accelerates

the computation due to the reduced degrees of freedom of the system. However, the angular velocity of the particles is still computed to consider the translational motion of the particles that is caused by rotation around a frictional contact.

The arbitrary angular shape of real sand grains generates a contact point between the grains that is eccentric to the centre of inertia and causes a resistance against rolling. In order to consider the resistance against rotation due to the real grain shape, a rolling resistance can be applied to the particles. Several rolling resistance models are already implemented in the used DEM software. Further approaches to consider the shape of the grains for geotechnical applications is the locking of rotation for a certain percentage of the particles or the generation of clumped particles by grouping several spherical particles to one irregular shaped particle. The influence of both, locking of rotation and using clumped particles, on the resistance of a cone penetration test was investigated by Falagush et al. (2015). It reveals that the locking of the particles' rotation results in an excessive large tip resistance and is an unsuitable approach to consider the real grain shape, whereas the use of clumped particles results in a tip resistance that depends on the clumped particle shape and can be used to model the physics of angular grains.

The rolling resistance models that are implemented in LIGGGHTS are a constant directional torque model and an elastic-plastic spring-dashpot model, which are explained in detail by Ai et al. (2011). In a paper of Jiang et al. (2015) a rolling resistance model is introduced considering a twisting resistance, which is implemented in a rolling resistance model that was additionally developed within this work.

The constant directional torque (cdt) model by Ai (2011) applies a constant torque on particles that acts always against the relative rotation in the contact of two particles. The torque is always applied to both particles, whereby a torque is transmitted from particle to particle. A problem that occurs using the cdt model is the oscillation of particles at rest position caused by the constant torque that is always alternating in direction. The oscillations of the particles produce a residual kinetic energy that destabilises the system and leads to a creeping in the macro-scale behaviour.

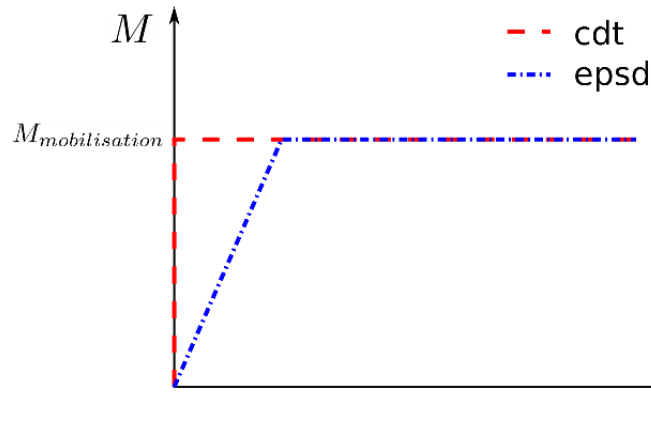


Figure 13: Spring characteristic curve for different rolling models

The elastic-plastic spring-dashpot (epsd) model solves the problem of oscillations at the rest position. Therefore, the model applies a rotational spring-dashpot to particles that are in contact. As soon as the spring force reaches a maximum value of resistance, the particle starts to rotate without any further increase of the resisting torque. The elastic part of the rotation is always recovered at unloading, which can cause an unphysical behaviour if the elastic part becomes too large. In the case of a high stress state, the restructuring due to changes in load is highly dependent on small rotations and movements of the particles. The deformation energy that is saved in the elastic springs of the rolling model will be recovered in deformation at unloading, causing an undesirable large rebound. An increase of the stiffness of the rotational spring can reduce this effect, but with increasing stiffness the model will start producing oscillations as it was observed for the constant directional torque model. The spring characteristic curve of both models is given in Figure 13.

Both the epsd and the cdt model apply a constant torque against the rotation of a particle as soon as the particle mobilises. This is physically based on the assumption of two spherical particles that overlap and create a flattened area in the contact. The contact point of the particle is shifted to an eccentric point due to the flattened area. The so created resisting moment is constant during rotation as long as the particle has a constant overlap. The rolling resistive moment at mobilisation $M_{r,plastic}$ is proportional to the normal contact force F_n and the particles effective radius r_{eff}

$$M_{r,plastic} = F_n r_{eff} \mu_r, \quad (4)$$

where μ_r is the coefficient of rolling resistance that has to be defined. This corresponds to a resistive moment due to an eccentric contact force with an eccentricity of $r_{eff} \cdot \mu_r$.

In geotechnical applications the resisting moment of a particulate material is less attributed to the overlap but rather to the non-sphericity of the grains. Estrada et al. (2011) investigated the bulk behaviour of polygons and spherical particles with a constant maximum rolling resistance in a shear test. They determined a possible mapping between the parameter for rolling resistance and the shape of the polygons by considering shear strength, solid fraction, force and fabric anisotropies. This indicates that a simple rolling resistance model can be used to imitate the effect of angular grains. But caution should be taken here for different loading cases.

Within this thesis, single particle simulations were investigated to obtain a more realistic rolling resistance model. Therefore, a linearised rough profile of the resisting moment of an ellipse and a cube, given in Figure 14, were implemented in LIGGGHTS and tested. The rolling resistance of an ellipse increases first due to the shift of the normal force out of the centre of the ellipse. At a certain point, the eccentricity of the normal force decays due to the tilt up of the ellipse. The resistance switches into an accelerating moment at 90 degree, when the ellipse is upright. However, the resistance of a cube has its maximum at the beginning of its rotation out of the rest position. The normal force acts at the edge of the cube with an eccentricity of half the edge length times the cosine of twice the rotation angle. Thus, the resistance decreases until the cube is on the edge with zero resistance. At this point, the resistance switches into an accelerating moment that increases again until the cube drops into its next rest position.

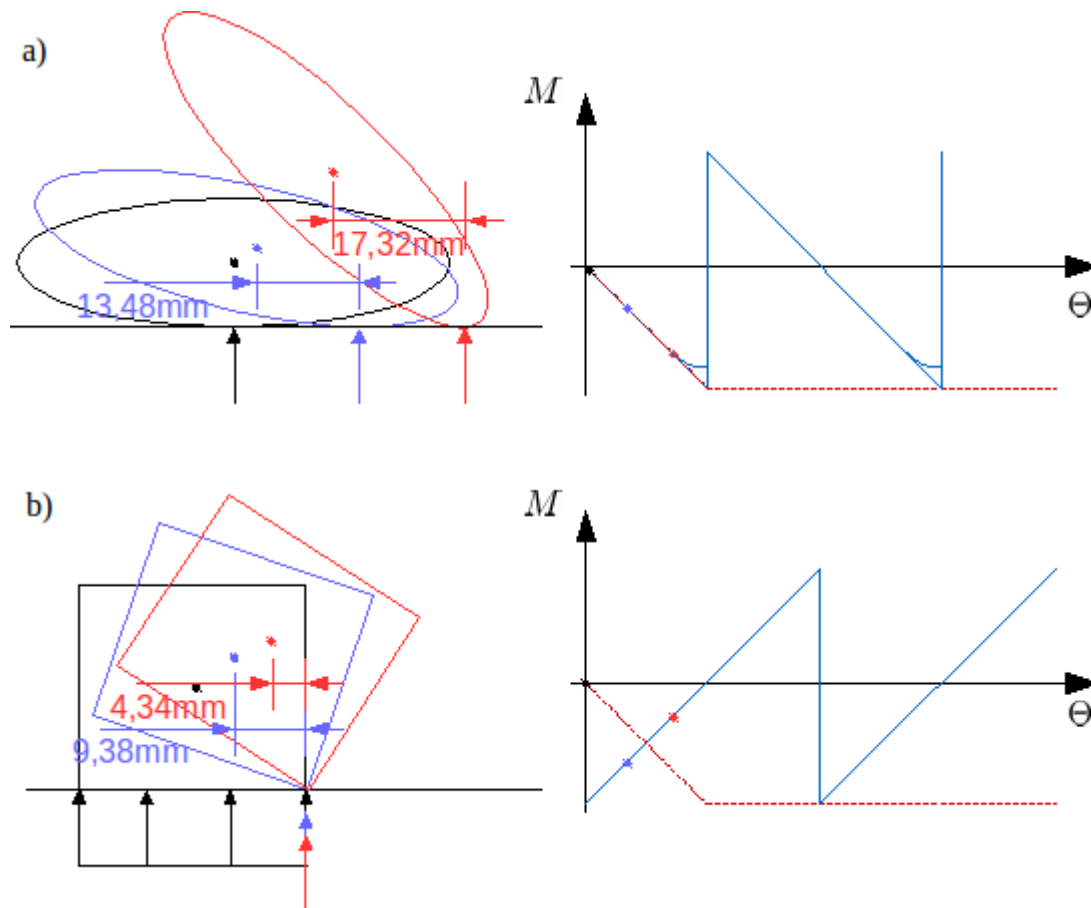


Figure 14: a) Rolling model for elliptical particles, b) Rolling model for angular particles

For the study on the rolling models, the roll over behaviour of an irregular shaped clumped particle was investigated and compared to the rolling behaviour from different rolling resistance models. The rotational and translational velocities of a particle rolling down an inclined plane were compared, applying different rolling resistance models, see Figure 15 & Figure 16. The applied rolling resistance over the relative rotation is shown on the right in both figures. It has been found that a resisting moment similar to the resistance of a cube results in a suitable behaviour for the case of a particle rolling down an inclined plane. This rolling resistance model was also tested in calibration tests, but it revealed that it was not stable enough for an adequate time step. Especially in an oedometer test, the particles began to oscillate under pressure.

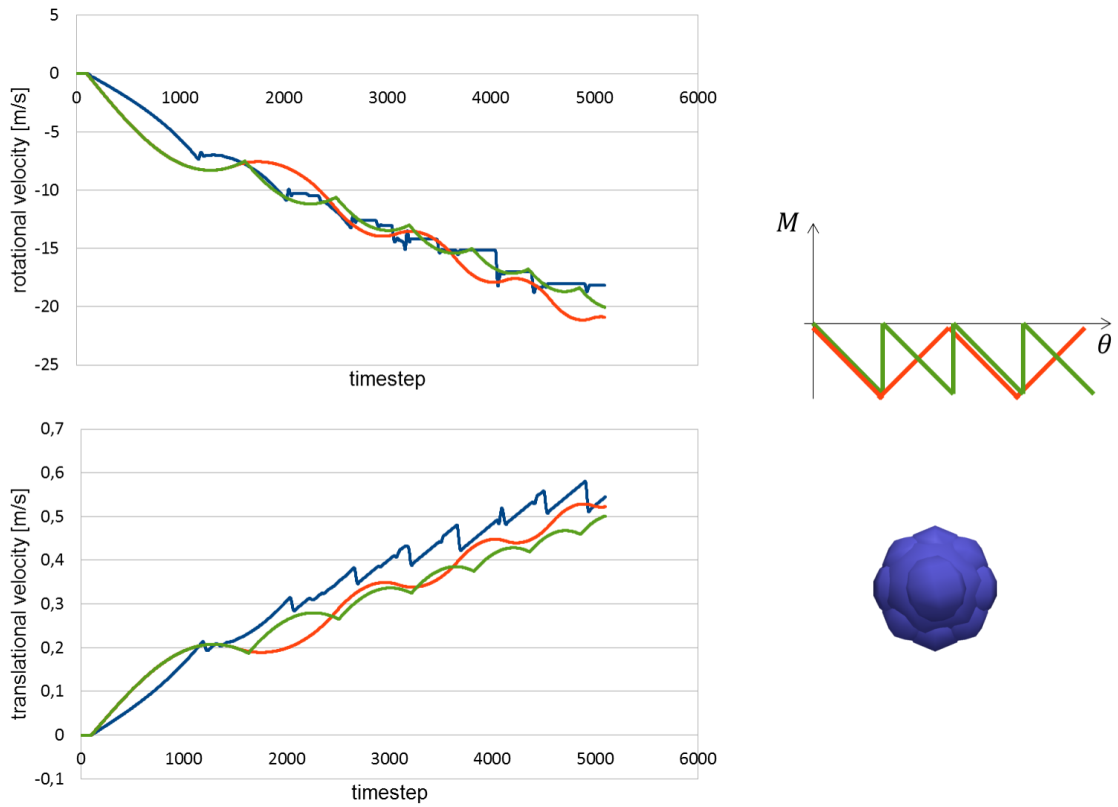


Figure 15: The rolling behaviour on an inclined plane of a clumped particle (blue) and a particle with a rolling resistance model similar to an ellipse (green & red).

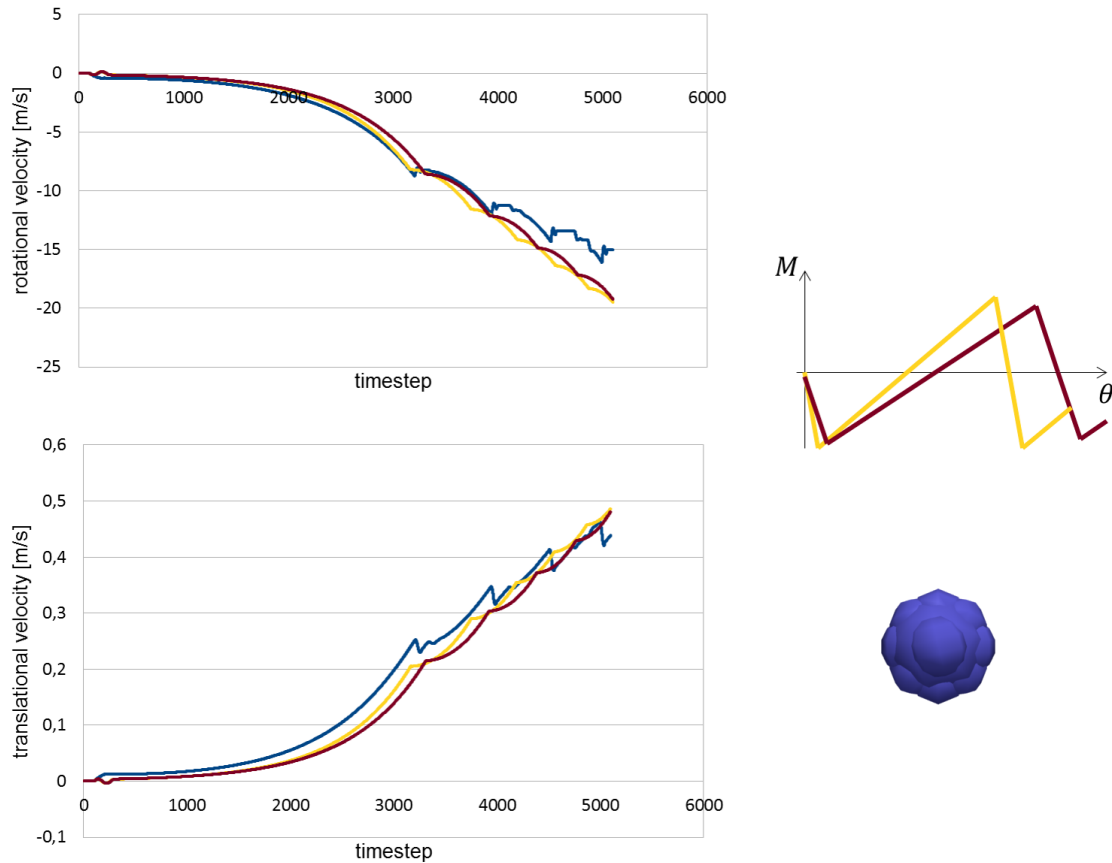


Figure 16: The rolling behaviour on an inclined plane of a clumped particle (blue) and a particle with a rolling resistance model similar to a polygon (yellow & purple).

For further simulations the simple elastic-plastic spring-dashpot model of LIGGGHTS was taken with modifications on the damping moment and the behaviour of the transition zone between zero and maximum resistive torque. The damping moment in the epsd rolling model acts during the elastic part and is not limited. The damping moment increases with relative rotational velocity of the contacting particles and may exceed the maximum torque at mobilisation. Thus, it is possible that an unphysical high rolling resistance is generated and the material would behave too stiff in the macro scale for a dynamic load. For this reason, the used rolling model is modified such that the total rolling resistance between two particles is always limited by the resisting torque at mobilisation.

Furthermore, changes on the rolling stiffness k_r of the epsd model were investigated. The default value for the rolling stiffness is given by

$$k_r = k_t r_{eff}^2 \quad (5)$$

with the tangential stiffness k_t and the effective radius r_{eff} defined as

$$\frac{1}{r_{eff}} = \frac{1}{r_1} + \frac{1}{r_2} \quad (6)$$

with r_1 and r_2 being the respective radii of the contacting particles. This leads to a certain rotation angle of the particles in the elastic region of the rolling model

$$\theta_{elastic} \propto \mu_r \frac{\delta_n}{r_{eff}} \quad (7)$$

with μ_r being the parameter that defines the interparticle rolling resistance.

In this way, the amount of elastic rotation $\theta_{elastic}$ depends on the overlap δ_n relative to the particles size. The problem that occurs using reduced elastic rotations is the necessity of a very small time step. However, using larger elastic rotations will cause larger elastic settlements and at some point also a weaker behaviour in strength, which was observed from simulations of triaxial tests. The weakening in strength is attributed to the separating of particles before they even reach their maximum rolling resistance.

The dependency of the elastic rotations on the particles overlap makes it difficult to develop a consistent algorithm to determine the resistive moment in the elastic part. It has to be considered that the tangential stiffness is dependent on the particles overlap as long as the Hertzian normal contact is used. Hence, also the rolling stiffness is overlap dependent. A change in the overlap during the contact of particles causes a change also in the rolling resistive torque, which has to be defined by an appropriate algorithm. The change of the resistive torque due to a change in the normal force is given by a new maximum torque and a different stiffness in the elastic region.

The rolling models that exist in literature for an explicit time integration scheme are described in Ai (2011), Wensrich (2012) and Jiang (2015). Ai (2011) named them a directional constant torque model, a viscous model, an elastic-plastic spring-dashpot model and further contact-independent models. The directional constant torque model and the elastic-plastic spring-dashpot model are explained in chapter 2.3, where they are named as the cdt and epsd model. The viscous model applies a torque to particles in contact that is proportional to the normal force and the angular velocities. The rolling resistance in the viscous model is just present as long as the particles are in rotation, while in equilibrium there will be no lasting resistance torque. The contact-independent models apply a resistive torque that is proportional to the particles respective angular velocity. Thus, the particles do not transmit a torque from particle to particle. The contact-independent models are not commonly used because of the unphysical approach behind it.

There exist also high sophisticated rolling models which are more practicable for implicit time integration schemes. A rolling model for the simulation of granular

material for geotechnical applications has been developed and implemented by Lichtenheldt (2013). The model was used for the simulation of sand grains under the load of a planetary rover wheel. The idea of the model is inspired by the resisting torque of rectangular geometries and the applied torque is dependent on the orientation of the particles, where each particle has its own function of resisting torque. Thus, it is possible to apply accelerating as well as resisting torques. The problem using this model with an explicit time integration schemes would be that the fast changes in the torque and the accelerating torque destabilise the system. Therefore, Lichtenheldt (2013) uses a semi-implicit Newmark integration scheme to solve the time integration with an adequate time step.

2.4 The Tangential Friction Model

A tangential friction is usually defined between two contacting bodies, while in a particulate material a network of many contacting particles exists. This friction from particle to particle causes an inner resistance of the material against shearing. A continuum parameter for particulate materials that specifies the resistance of the material against shearing is the internal friction angle. This parameter depends on the interparticle friction as well as on the rolling resistance, which reflects the shape of the grains. Therefore, the DEM applies a tangential friction model to the particles in addition to the rolling resistance.

The tangential friction model in LIGGGHTS is very similar to the rolling resistance model from chapter 2.3. It consists of an elastic and a plastic part with an additional damping component within the elastic part. The damping force depends on the relative tangential velocity of the contacting particles. This damping force is not limited in the original source code of LIGGGHTS. Therefore, a modified version of the tangential friction model has been developed to limit the total tangential force always by the Coulomb friction force. A schematic of the mechanical principle of the model and the spring characteristic curve are shown in Figure 17.

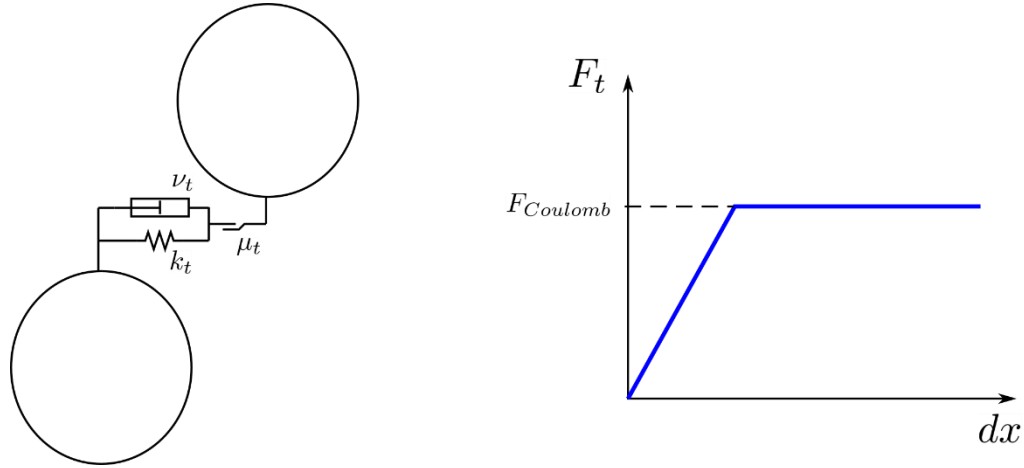


Figure 17: Mechanical scheme and spring characteristic curve of the tangential contact model

The tangential friction force F_t increases linearly with the relative movement dx of the contacting particles until it reaches the Coulomb friction force $F_{Coulomb}$. At this point the particles start to mobilise under a constant tangential friction. The friction force in the elastic region and the limiting Coulomb force in the plastic region are defined as follows:

$$\begin{aligned} F_t &= k_t \cdot dx \\ F_{Coulomb} &= F_N \cdot \mu_t, \end{aligned} \quad (8)$$

where the tangential stiffness k_t defines the elastic region. If the tangential stiffness is too large, the model becomes unstable and the time step needs to be reduced. Otherwise, if the tangential stiffness is very low, the elastic part of the relative movement of particles is increased and leads to an unphysical behaviour. Therefore, the tangential stiffness is set to be as stiff as possible keeping a stable simulation with a time step that has been defined by the normal contact model. The default value of the tangential stiffness in case of the Hertzian normal contact is given by

$$k_t = 6 \frac{G}{Y} k_n \quad (9)$$

with the Young's modulus Y and the Shear modulus G .

For the determination of the coefficient of tangential friction μ_t and rolling friction μ_r , it has to be identified first which type of motion is dominant. It becomes clearer by comparing the accelerating torque of a particle on an inclined plane due to the tangential friction and the rolling resistive torque due to the eccentric normal force.

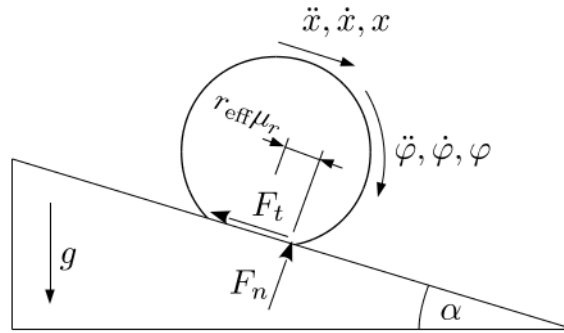


Figure 18: Mechanical scheme of a particle on an inclined plane

The moments that act on the particle centre on in inclined plane are an accelerating torque due to the tangential friction in the contact and a rolling resistive torque due to the eccentric contact force, see also Figure 18. The equations of motion for translation x and rotation φ of a particle are

$$m\ddot{x} = -F_n\mu_t + mg \sin \alpha \quad (10)$$

and

$$I\ddot{\varphi} = F_n\mu_t r - F_n\mu_r r_{eff} \quad (11)$$

with the particle mass m , the rotational inertia I and the gravity g . In the case of a particle-plane contact, the effective radius r_{eff} becomes equal to the particle radius r and equation 11 can be transposed to

$$I\ddot{\varphi} = F_n r (\mu_t - \mu_r). \quad (12)$$

These equations just hold for a particle that is mobilised in rotation and translation in positive directions. The equation of motion for translation can be transformed with

$$F_n = mg \cos(\alpha) \quad (13)$$

to

$$\ddot{x} = g(\tan \alpha - \mu_t) \quad (14)$$

and determines the angle α of the slope that is necessary for the sliding of the particle

$$\tan \alpha > \mu_t. \quad (15)$$

If the angle of the slope is too low to provoke the sliding of the particle, it may be possible that the particle starts to rotate instead. The tangential friction for a sticking particle is equal to the downhill force due to the force equilibrium. This results in an equation of rotational motion for the particle of

$$I\ddot{\varphi} = F_n r \tan \alpha - F_n \mu_r r_{eff}. \quad (16)$$

For this case the particle starts to rotate as soon as

$$\tan \alpha > \mu_r. \quad (17)$$

is fulfilled. This means that if the coefficient of rolling resistance is smaller than the coefficient of tangential friction, the particle would start to roll down the plane rather than sliding and vice versa.

This has to be considered for the choice of the coefficients of rolling and sliding resistance. Therefore, it has to be decided whether the particles are highly angular and would rather slide than rotate or if the particles are assumed to be more round.

2.5 The elastic-plastic yield criterion for frictional contacts

The algorithm for the rolling and twisting resistive torques as well as for the tangential friction is a modified version of the algorithm that is already used for the tangential friction in LIGGGHTS. The current resistive value is computed by the accumulated amount of elastic deformations times the current stiffness of the corresponding model. The increasing resistive value is limited by a maximum resistance and the particles will begin to mobilise as soon as the limiting resistance is acting. The algorithm consists of two parts, the elastic part, where all deformations are recovered at unloading, and the plastic part, where permanent deformations are generated. The plastic part is triggered by the exceeding of the maximum resistance. An additional damping is applied within the elastic part to reach a stable position without large oscillations. The total value, that is the sum of resistance and damping, is also limited by the maximum resistance.

The elastic-plastic yield criterion for the frictional contacts in LIGGGHTS behaves fully elastic until the limiting value for the plastic phase is reached. Thus, it can happen that a large amount of deformation can be stored in the elastic phase that will be recovered at unloading. This problem can be observed in the oedometer test at unloading and reloading, where the soil does not compact as desired. Furthermore, the elastic rebound takes a large part of the settlements during a penetration cycle of HP³ and causes unrealistic large rebound after each penetration stroke. For this reason, improvements of the yield criterion are made to reduce the amount of stored elastic deformation. A new algorithm that is implemented applies different stiffnesses for the elastic phase at loading and unloading/reloading. Thus, lasting settlements will also occur in the elastic phase due to a stiffer behaviour at unloading. A switch from unloading/reloading to primary loading is triggered by a change in the relative displacement direction.

For the implementation of the new algorithm, two state values are necessary to describe the torque or force level. The first value is the elastic relative displacement and the second value is the sum of the elastic and plastic relative displacements, see Figure 19.

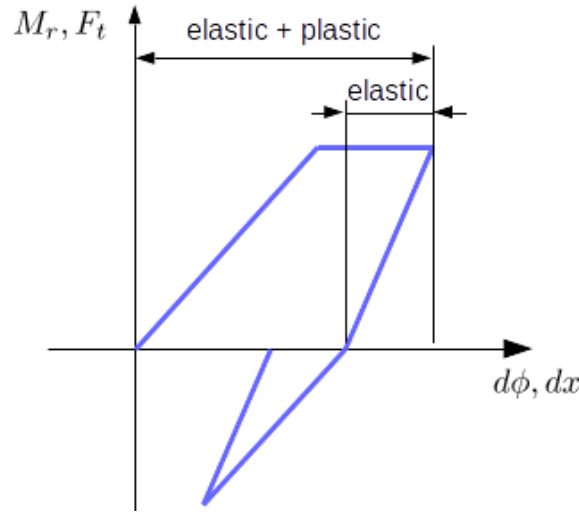


Figure 19: Sketch of the yield algorithm with different stiffnesses for primary loading and un-/reloading.

The maximum resistive force increases linearly under primary loading until the Coulomb friction force $F_{Coulomb}$ is reached. The displacements at primary loading consist of elastic dx_e and plastic deformation dx_p and the limiting resistive force $F_{t,max}$ depends on the sum

$$F_{t,max} = \begin{cases} k_{t,prime}(dx_e + dx_p), & F_{t,max} < F_{Coulomb} \\ F_{Coulomb}, & F_{t,max} > F_{Coulomb} \end{cases} \quad (18)$$

with the stiffness $k_{t,prime}$ for primary loading. The total shear dx_{e+p} is adjusted if the Coulomb friction force is reached:

$$dx_{e+p} = \begin{cases} dx_{e+p}, & F_{t,\max} < F_{Coulomb} \\ F_{Coulomb}/k_{t,prime}, & F_{t,\max} > F_{Coulomb} \end{cases} \quad (19)$$

The stiffness for the computation of the resistive force is increased at unloading. Therefore, even a loading-unloading cycle below the Coulomb friction lasts in settlements, as it can be seen in Figure 19 during primary loading in the negative direction. The limiting resistive force is also reduced again with relative displacement in the opposite direction, so that the particle can move again under primary loading.

The algorithm uses two more state variables to save the preloading of the particle at a reversal of the movement direction and to reduce the preload with further movement in the reverse direction. For this purpose, the plastic displacement vector at direction reversal is stored in a variable dx_{stored} and the total shear that takes place starting from the last direction reversal is saved in a variable $dx_{\Delta shear}$.

At the beginning, the elastic shear is updated for time step $k+1$ with step size dt and the friction force is calculated for the case of an un-/reloading behaviour

$$\begin{aligned} dx_e^{k+1} &= dx_e^k + v_t^k dt \\ F_t &= k_{t,un/re} dx_e^{k+1} \end{aligned} \quad (20)$$

with the un-/reloading stiffness $k_{t,un/re}$. If the plastic shear vector dx_p^k and the elastic shear vector dx_e^{k+1} point into the same half space, there is no direction reversal and the total shear and $dx_{\Delta shear}$ can be updated by

$$\begin{aligned} dx_{e+p}^{k+1} &= dx_e^{k+1} + dx_p^{k+1} = dx_{e+p}^k + v_t^k dt \\ dx_{\Delta shear}^{k+1} &= dx_{\Delta shear}^k + v_t^k dt, \end{aligned} \quad (21)$$

otherwise a motion reversal is triggered. In the case of a motion reversal, the current plastic shear displacement will be stored in dx_{stored} and set to the sum of the old value of dx_{stored} and the unidirectional shear displacement $dx_{\Delta shear}$. The unidirectional shear displacement is the plastic shear that occurred since the last motion reversal. If the plastic shear displacement is pointing into the same half space as before, it will be reset to the current elastic shear dx_e . The value of $dx_{\Delta shear}$ is set to zero at motion reversal, so that it starts counting on from the current reversal point.

The maximum resistive force $F_{t,\max}$ is updated by equation 18 for each time step. The friction force F_t is compared to $F_{t,\max}$ and adjusted, if F_t exceeds the value of $F_{t,\max}$:

$$F_t = \begin{cases} k_{t,un/re} dx_e^{k+1}, & F_t < F_{t,\max} \\ F_{t,\max}, & F_t > F_{t,\max} \end{cases} \quad (22)$$

Furthermore, the elastic shear displacement is adjusted if the maximum resistive force is reached:

$$dx_e = \begin{cases} dx_e, & F_t < F_{t,\max} \\ F_{t,\max} / k_{t,un/re}, & F_t > F_{t,\max} \end{cases} \quad (23)$$

In the end, a damping force is added to stabilise the oscillating system. The total force $F_{t,total}$ in the frictional contact is always limited by the Coulomb friction force:

$$F_{t,total} = \begin{cases} F_t + F_{damping}, & F_{t,total} < F_{Coulomb} \\ F_{Coulomb}, & F_{t,total} > F_{Coulomb} \end{cases} \quad (24)$$

2.6 Validation of the Contact Models

The contact models in the DEM are necessary to consider the physics of the granular material that has to be modelled. For the application to geotechnical problems, the explained contact models in chapters 2.2, 2.3 and 2.4 have to be considered. The used contact models have to be validated in terms of a correct physical behaviour and numerical stability. Therefore, the contact forces and torques have to be related to physical quantities. For the validation regarding the numerical stability, the motion of the particles during different kind of simulations have to be investigated. Numerical instabilities often provoke high kinetic energies which can be missed if they appear only in a few particles. The translational and the rotational motion of particles are investigated in simulations with a single particle as well as using multiple particles to figure out if instabilities occur and to identify the origin.

The models that are investigated for the validation of the contact models are a single particle on an inclined plane, a particle rolling in a pipe as well as simulations of colliding particles (Figure 20). Furthermore, the stability of a particle package using a few thousand particles is tested to prove the overall behaviour.

The inclined plane model is used for investigations on the stability of the contact model and to verify the general behaviour. The particle rolling in a pipe up and down is used to observe the contact behaviour at a change in the direction of movement and to validate the energy dissipation. Further simulations of many particles are used to determine the behaviour of the contact model at colliding of particles as well as the force and torque transmission. The oscillations of the particles motion as well as the force and torque are investigated to identify any irregularities in the contact behaviour.



Figure 20: Test models for the validation of the contact models.

The tangential and the rolling resistance depend also on the normal contact force between the particles. Therefore, the algorithm for the calculation of the resistive force or torque needs to be verified for all possible cases regarding a change in the contact force. The sketch in Figure 21 represents a possible pathway of the tangential friction force or the rolling resistance torque, where a change of the force or torque due to an increasing contact force is displayed in blue lines and due to a decreasing contact force in green lines. The rotation or translation starts always in the elastic region of the corresponding model, where the force or torque increases linearly with a movement. An increase of the contact force leads always to a jump of the resistive force/torque into the elastic region with a larger stiffness and maximum resistance. Where a decrease in the contact force will result in a jump into the elastic or plastic region with a decreased stiffness and maximum resistance. It can be seen in Figure 21 that at the first decrease of the contact force (first green connection) the model jumps from the elastic into the plastic region, while at the second decrease in the contact force there is a jump in the resistance but it stays in the elastic region. The standard EPSD2 rolling model of LIGGGHTS uses an algorithm that can cause backwards rotation due to a change in the contact force even if the torque was just applied in one direction. This behaviour is improved in the modified version that is used for the simulations. Therefore, the resistance is calculated based on a total relative displacement and not changed by increments. Thus, the applied algorithm controls the resistance in a way that lasting settlements of particles will only occur in the direction of the acting torque or get back into the initial position.

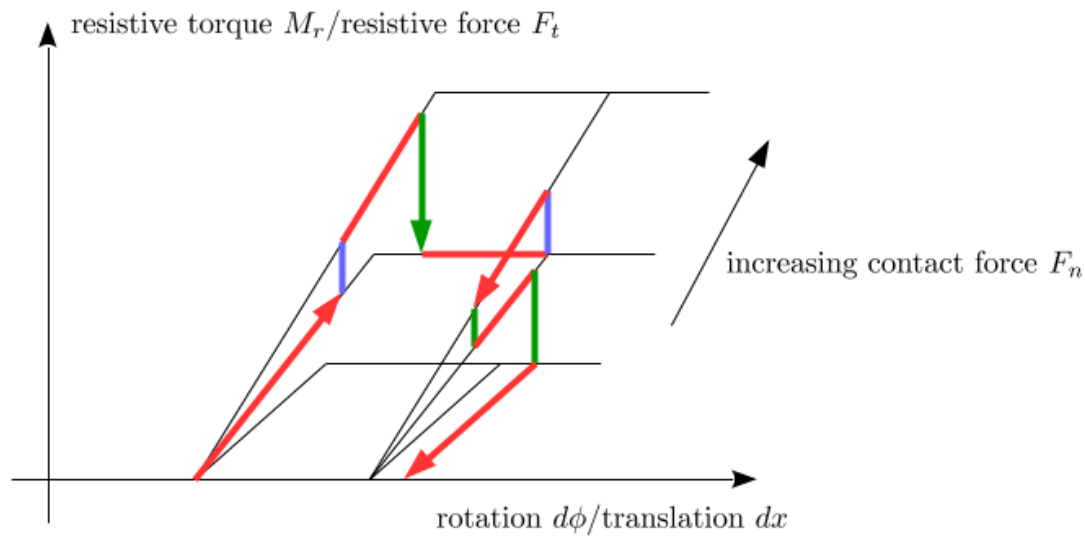


Figure 21: A graphical sketch of the algorithm for the tangential and rolling model

2.7 Neighbor lists

The neighbor lists specifies the particles for the contact computation, see LIGGGHTS®-PUBLIC documentation (2017). The preceding determination of neighbouring particles is necessary to reduce the amount of contact computations. For this purpose, the pairs of particles that will not interact in the next few time steps are neglected in the force computation. Thus, it limits the amount of contact computations to particles that are close to each other. The neighbouring particles in a certain distance are determined and checked for possible force interactions. The skin distance defines the domain to search for neighbouring particles and can be set manually by the *neigh* command, where the skin distance is the additional space between particles before they get in contact, see Figure 22.

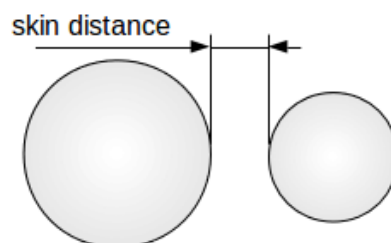


Figure 22: Skin distance to search for neighbouring particles

A smaller skin distance causes the code to rebuild the neighbor lists more often but the number of computed contacts is less, while a larger skin distance increases the

number of contacts that have to be computed and reduces the amount of neighbor lists.

The skin distance should be set by default to the maximum radius of the used particles, but can be reduced for simulations with less motion. In the case of a wide range of different particle sizes, many neighbouring particles would be found if the skin distance is related to the largest particle radius. In this case, it has to be judged which skin size is the best fit.

In the case of the penetration simulation, where four different domains of scaled particles sizes exist, the skin distance is chosen related to the maximum radius of the particles in the core domain. In the filling phase the neigh size is set equal to the corresponding radius and is reduced to half of it after the particles have settled, since the particles motion is very slow from this point onwards.

2.8 The initial filling process

The initial filling process generates a certain density in the soil specimen and creates the initial stresses in the soil. Furthermore, the filling can cause a sorting of grains due to the granular segregation or compact the soil locally due to the drop height of the particles. Hence, a particle radius expansion method is used to prepare a homogenous soil bedding (Bernhardt et al. 2015).

For the particle radius expansion method the particles are inserted with a smaller radius than they will have later in the simulation. The initial particle volume for the insertion is decreased by the particles reduction scale to the power of three, whereas the total volume of the filling area is kept constant. Thus, a very loose packing is generated first and the insertion without an overlap of particles is easier. The insertion is done in a zero gravity environment so that the particles hover inside the filling domain. After the insertion is done, the particles radii are expanded stepwise up to their desired size. The interparticle friction and rolling resistance is kept zero, and the Young's modulus is reduced until the particles reached their desired radii and stopped moving. Then, the particles' parameters for the simulation are applied and the gravity is turned on.

The settlement of the soil particles and the overburden particles is performed in parallel. The overburden particles are inserted in a close distance to the upper boundary of the soil domain, so that the overburden particles generate less kinetic energy.

The creation of a desired packing density is difficult to achieve in a particle code. One option is the choice of the particles insertion domain such that the particles compact during the radii expansion process. The loose particle bedding for the simulations is prepared in a domain that is even larger than necessary to avoid a

close arrangement of particles. Due to the spherical shape of the particles it is not always possible to achieve the loosest packing of a granular material. However, the generated density can be measured in an accurate manner by the computation of the volume and mass of a Voronoi tessellation for an embedded region of particles.

3 Calibration

The calibration of geotechnical materials in the DEM is usually done by investigations of the macro scale behaviour and adjustment of the micro scale parameters (O’Sullivan 2011), such as interparticle friction. These particle parameters does not directly correlate with continuum material parameters but can be adjusted to produce the same mechanical behaviour. Therefore, all tests are modelled with the DEM and the system response is compared to the laboratory measurements. The particles’ parameters that have to be defined are:

- Young’s modulus and poisson’s ratio for soil stiffness
- Tangential friction and rolling resistance for soil strength
- Coefficient of restitution for damping.

For the sake of calibration, different soil tests has been investigated, where a Martian analogue material is used to reproduce a similar soil behaviour as it is supposed to be at the InSight landing site. The soil tests that are available at the laboratory of the Institute of Soil Mechanics and Foundation Engineering are an angle of repose experiment, an oedometer test and a triaxial shear test. The angle of repose experiment and the triaxial shear test provide information on the inner shearing resistance of the soil, whereas in the oedometer test the stiffness of the material can be determined. The strength of soils is often stated by its inner resistance against shearing, as long as no grain crushing is involved. This shear resistance depends on the grain to grain friction and the grain specific rolling resistance. In geotechnics it is common to describe the shearing resistance by the internal friction angle and the dilatancy angle. Instead, in the DEM the specific values for grain to grain friction and rolling resistance are applied and dilatancy effects are automatically captured within the simulation.

3.1 Material

The material that is investigated in this work is a local soil mined in Austria that has a similar grain size distribution as the known Martian simulant JSC-Mars 1. It is a sieved quartz sand smaller than 1 mm denoted as “Schwarzl UK4”. The Schwarzl UK4 has already been used as a Martian simulant for penetration simulations at the Space Research Institute in Graz, Austria (Zöhrer 2006). The gradation of the particle size is very uniform and has a uniformity coefficient C_U of 4, which can be well reproduced in DEM. The particles’ size is upscaled in the simulations depending on the model. A simple element test, for example, can be modelled using only a few thousand particles without much difference in the results. The characteristic of an element test is a homogeneous stress distribution, e.g. in a uniaxial compression test. The maximum scale of the particle size is kept

small enough to avoid the formation of long force chains carrying most of the load. The material has an internal friction angle of about 33° and no cohesion. The bulk density ranges from 1300 to 1700 kg/m³ with a grain density of 2700 kg/m³. Thus, the void ratio is 0.58 for the densest packing and 1.07 for the loosest packing. The grading curve of Schwarzl UK4 is shown in Figure 23. For the implementation in the DEM the grading curve is adjusted (yellow line). The small amount of very large particles and very small particles is neglected. The used particle size distribution is generated out of values between the grading sizes. Instead of only using the mean value between the grading sizes (blue line), a uniform distribution of particle radii (green line) is applied to get a homogeneous soil behaviour.

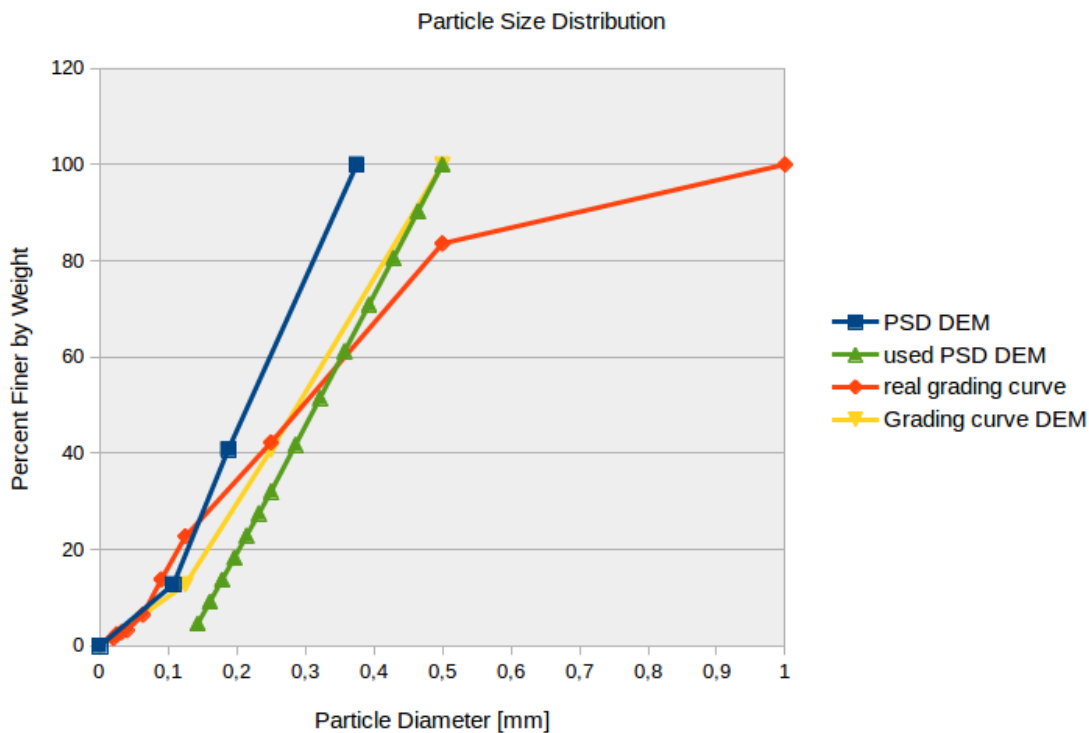


Figure 23: Particle size distribution of Schwarzl UK4

3.2 Angle of Repose

The angle of repose experiment takes only little effort and provides first information about the internal friction of the material. It is a common test used for dry cohesionless granular materials. This test creates a natural slope by lifting up a hollow tube filled with granular material, see Figure 24. The filling height of the tube has to be large enough, so that the critical slope angle (i.e. the largest possible slope angle) can be achieved. For a cohesionless dry sand the natural slope angle corresponds directly to the internal friction angle at critical state. The critical state means that shearing occurs at constant volume and that interlocking of grains does not affect the strength. This condition is present for normally consolidated soils or for overconsolidated soils at large shear strains, when softening has taken place.

The advantage of the angle of repose experiment is the minor influence of the soil stiffness on the obtained results. Thus, it is possible to derive the friction parameters without adjusting the stiffness parameters in the simulation model. Since in the DEM the interparticle friction and the rolling resistance define the inner shearing resistance, both parameters need to be adjusted to match the right slope angle. The outcome of this experiment is not a specific set of friction parameters but a series of sets that reveal a reasonable slope angle. For this reason, it has to be specified beforehand if the grains are more rounded or angular which corresponds to a higher interparticle friction or rolling resistance, respectively.

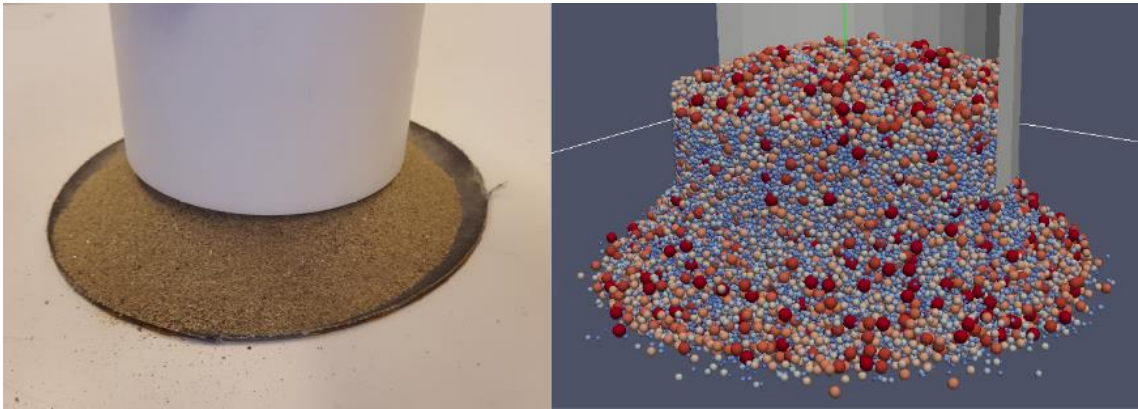


Figure 24: Generation of a critical slope by lifting a filled cylinder.

An algorithm is applied to evaluate the angle of repose in the simulation in an automatic way. The algorithm divides the sand pile in n horizontal slices and determines the maximum and minimum positions of the particles in the horizontal x - and y -axis in each slice, where the centre of the pile is at zero position (Figure 25). The top and the bottom slices are neglected to avoid errors due to a flattened tip at the top or wide spread particles at the bottom. The maximum pile diameter in each slice is calculated by the difference of the maximum and minimum position of particles in the horizontal axes from each slice. The ratio of the height of a slice to the difference of the maximum pile radii from slice to slice yields the inclination of the pile. Thus, it is possible to automatically evaluate the slope angle for many runs using different sets of parameters.

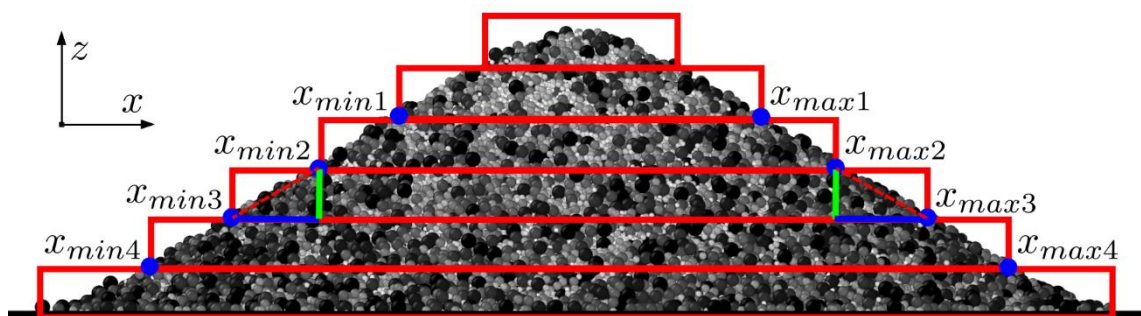


Figure 25: Sketch of the algorithm to determine the slope angle

In the simulation model, the sliding friction and the rolling resistance between the particles and the base plate is set to a value of 2.0 which is always larger than the interparticle friction and rolling resistance. In the lab experiment a sand paper is used as a base plate to create a large friction between the sand grains and the base plate to avoid the influence of slipping at the base. Furthermore, a teflon tube is used to reduce the friction between particles and the lifted casing, wherefore in the simulation model the friction of the confining material is set to zero. A low friction at the lifted casing ensures a continuous outpouring of the material.

3.3 Triaxial Shear Test

The triaxial shear test is an element test that fails a probe at different effective mean stresses to determine the strength of a granular material. Therefore, a cylindrical chamber is filled with saturated material and consolidated by applying a confining pressure. The lateral pressure is generated by using a membrane for the horizontal boundary and a surrounding fluid. The probe is then sheared under a constant lateral pressure. The shear rate is chosen slow enough to avoid the development of excess pore water pressure. The major principle stress during shearing is measured on the top and bottom wall while the minor principle stress at the membrane is kept constant. The triaxial test provides information on the peak resistance and the critical state strength of the material. The peak strength is the maximum resistance of a granular material during a shear test. However, the strength at critical state is defined when ongoing shearing occurs under constant volume. The interlocking of grains due to a dense bedding causes the material to expand due to shearing and produces a peak resistance that is reduced again by further shearing. The peak resistance depends on the void ratio and is related to the dilative behaviour of the soil skeleton, whereas the critical state strength is independent on the initial void ratio.

With a DEM model of the triaxial test, not only the strength values can be determined, but the stress strain relationship can be evaluated and used for the calibration. For the implementation in the DEM, the cylindrical boundary is approximated by six planes forming a hexagonal prism, see also Figure 26. An additional wall at top and bottom completes the chamber and hold the particles inside. The pressure on each wall is measured and controlled by a movement of the walls using the *fix mesh/surface/stress/servo* command of LIGGGHTS. The controller uses the error between the set-point and the actual value as well as the rate of change to approach the required pressure. This so called PD-controller acts very rapidly on stress changes, which reduces the computation time and it turned out to be the most stable choice for this application. The controller parameters were determined by a script of Abel (2010), where the proportional constant K_P is set to 7.5 % of the confining pressure and the differential constant K_D is set to

$$K_D = 0.12K_p dt \quad (25)$$

with the time step size dt of the simulation.

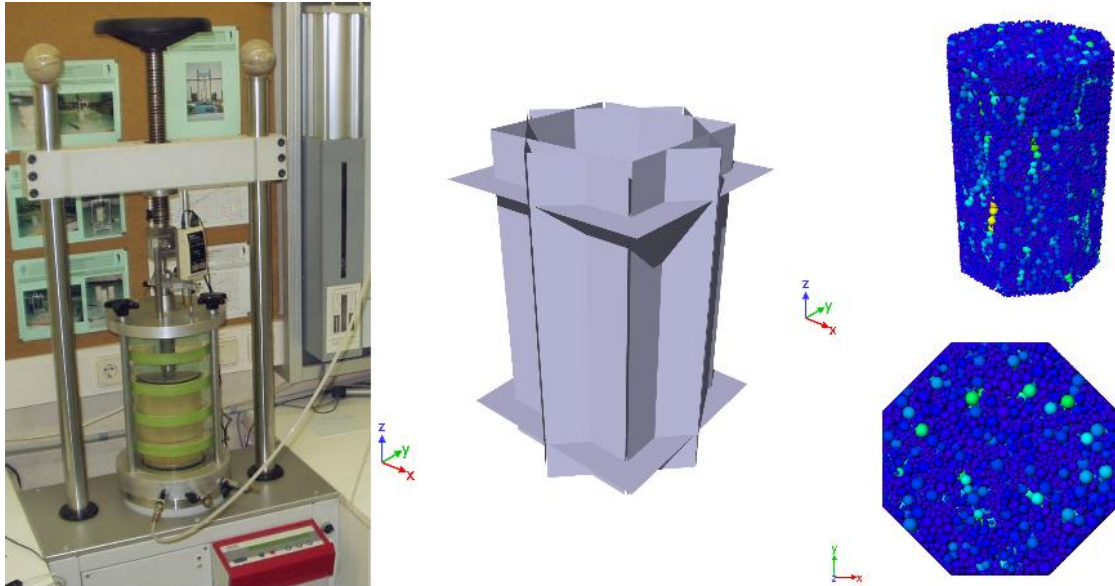


Figure 26: The triaxial shear test at the laboratory on the left (Zöhrer, 2006). The hexagonal prism chamber as well as the particulate material in the model, from perspective and top view, on the right.

After the consolidation phase, the top wall is locked while the bottom wall is moved upwards to shear the probe. The shear velocity in the DEM simulation can be much faster as in the lab test, since there is no water considered in the simulation that could generate excess pore water pressure. The vertical load acting against the bottom wall is measured and evaluated to determine the maximum resistance and the critical state strength. The volume of the probe is measured by the positions of the surrounding walls. The computation of the stress at each wall is implemented in the *controlforces* file. The contact area of each wall with the specimen is updated throughout the simulation to compute the stress out of the forces. Three simulations are carried out using a confining pressure of 100, 150 and 200 kPa respectively. The maximum shear stress at different mean normal stresses reveals a failure envelope that increases almost linearly with the mean normal stress. The inclination of this failure envelope is defined as the internal friction angle of the material. The internal friction angle at critical state corresponds to the angle of repose for a cohesionless dry sand. Thus, the triaxial shear test should result in a similar angle than the angle of repose experiment.

3.4 Oedometer Test

The oedometer test is a uniaxial compression test with lateral confinement that focuses on the stress strain relation to determine the stiffness of a material. The

bulk stiffness of granular materials is non-linear and depends on the current stress state as well as on previous stress states that could have compacted the material in the geological history. The oedometer test in the lab generates a static pressure in the sample using a stamp at the top of the probe while the settlements of the stamp are measured to derive the stress strain relation, see Figure 27. The material of the confining elements has to be stiff enough to avoid their deformations during the load application. The stress path is applied in increments that always doubles and begins with a stress of 10 kPa. After a stress of 320 kPa is reached at primary loading, the probe is unloaded first to 80 kPa and finally to 20 kPa. The load is again increased stepwise by doubling up to a value of 640 kPa. This stress path allows to investigate the primary loading as well as the unloading/reloading behaviour.

The triaxial and the oedometer test are both element tests, which means that there is a homogeneous stress state in the probe. In comparison to the triaxial test, where a constant lateral pressure is applied, the oedometer test has a confining ring to constrain the lateral strain of the probe. This boundary condition inhibits the soil to fail by shearing.

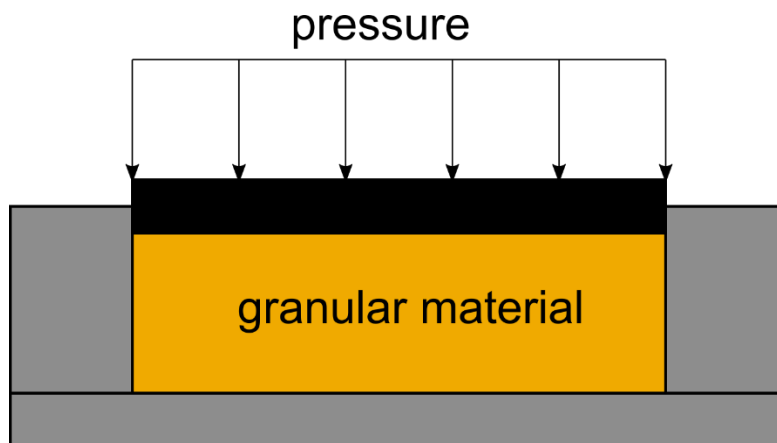


Figure 27: A sketch of the oedometer test.

The oedometer test is implemented in LIGGGHTS using a fixed cylindrical wall for the horizontal boundary and a fixed plane for the bottom. The top wall can only move vertically and is stress controlled using the *fix mesh/surface/stress/servo* command. The stress controlled top wall of the oedometer is used to apply different stresses. To avoid large oscillations in the stress path, the proportional constant for the controller is set much lower than for the triaxial test. A value of 2.4 for the proportional constant K_P was found to be an appropriate solution, whereas the differential constant K_D was set to

$$K_D = 0.2K_P dt . \quad (26)$$

The controller that drives the top wall reacts much slower and it occurs no overshoot of the wall displacement during compaction, which could compact the material and affect the soil stiffness. The force on the top wall and the corresponding strain is measured and used to determine the stiffness of the material. The assembled sample of the oedometer test in the DEM is shown in Figure 28.

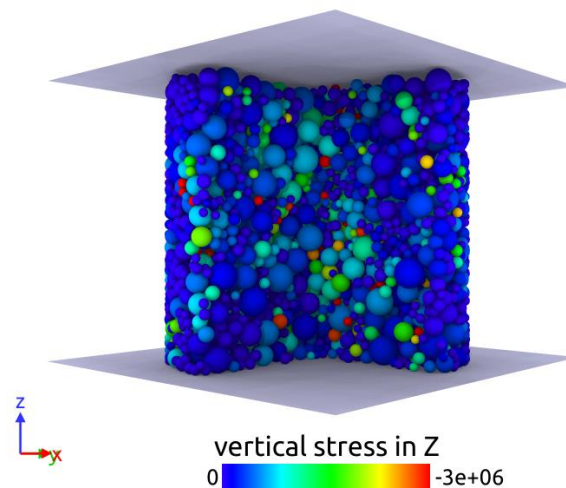


Figure 28: A cut view of the DEM model of the oedometer test with a colour coding of the vertical stress in Pa.

The oedometer test is done using a loose sample under dry conditions for the simulations as well as for the lab tests. The test is mainly used to determine the particle stiffness for the DEM by fitting the bulk stiffness of the oedometer test, where the lowest possible particle stiffness is preferred. The reason for a low particles stiffness is the related time step, which increases with smaller stiffness and thus less computation time is needed.

The stiffness of a granular material depends mainly on the structure of the soil skeleton and is independent of the particles weight for the case of a static load. Thus, it is possible to artificially increase the density of the particles and decrease the gravity, such that the overall stress state is the same. Due to the increased density of the particles, the system reacts slower and the time step can be increased. This technique is called mass scaling and is just applicable as long as it is a quasistatic simulation.

3.5 Calibration Results

The three calibration tests reveal parameters for the interparticle contacts to be able to model the strength and the stiffness behaviour of the investigated Schwarzl UK4 sand. For the calibration procedure, the Hertzian normal contact, a modified

tangential history model and a modified version of the EPSD2 rolling model of LIGGGHTS are used and result in the interparticle parameters listed in Table 2.

Table 2: Particle parameters for Schwarzl UK4 sand

Young's Modulus	0.8 GPa ($8 \cdot 10^8$ Pa)
Poisson's ratio	0.3
Coefficient of restitution	0.1
Coefficient of friction	0.6-0.7
Coefficient of rolling friction	0.4-0.5

With these particle parameters, the angle of repose experiment results in an angle of about $33^\circ \pm 1^\circ$. The direct comparison of the simulation and the experimental result can be seen in Figure 29. There is a good agreement in the shape of the pile that is created by pouring the material, where the angle of the slope corresponds to the critical internal friction angle of the material.

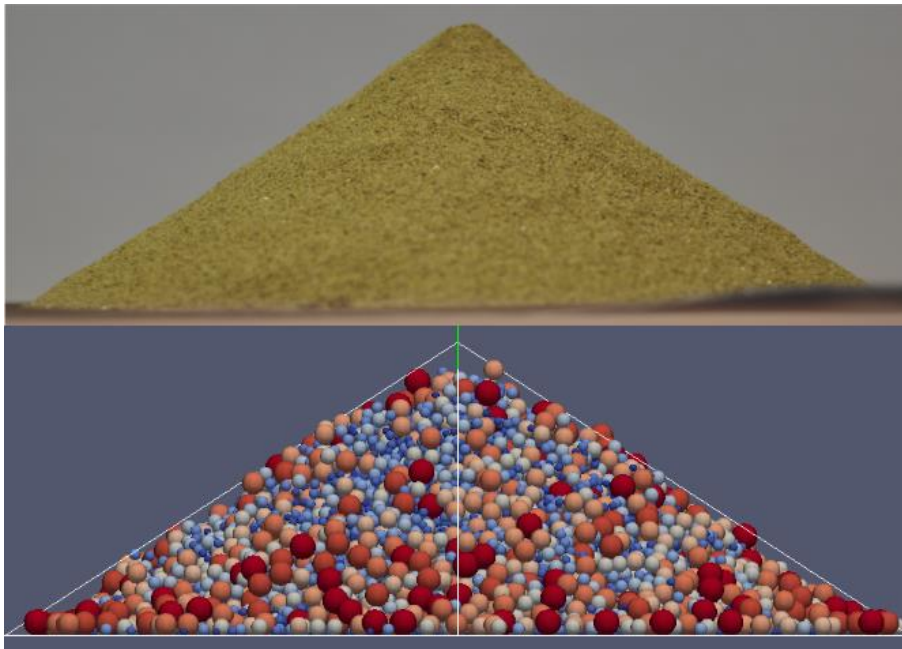


Figure 29: Angle of repose from experiment (top) and simulation (bottom)

The simulation of the triaxial shear test confirms the critical friction angle of about 33° and the comparison of the stress-strain relationship during the shear phase gives additional information on the soil stiffness. A secant stiffness modulus can be determined that defines the stress-strain relation of the material at primary loading and depends on the mean normal stress. The value of this modulus is

determined by a secant through the point at 50 % of the peak strength and is therefore also dependent on the initial void ratio. The comparison of the stress-strain curve of the triaxial shear tests for lateral stresses of 100, 150 and 200 kPa is shown in Figure 30. The stress strain curve at 100 kPa confining stress reveals that the critical and the peak strength of the material as well as the secant stiffness modulus can be well modelled in the DEM. The lab tests for 150 kPa and 200 kPa differ from the DEM simulations in stiffness for 200 kPa and in strength for 150 kPa. Nevertheless, to prove the correctness of the DEM results, the shear stress over the mean normal stress is plotted in Figure 31. The black lines represent the critical state friction angle of 32° and the peak friction angle of 35° . The peak value of each triaxial test is marked with a point and the critical state value with a square. It is noticeable that the critical state strength illustrated by squares fits almost perfect on the line of 32° friction angle for the DEM results. Furthermore, the peak strength that are marked by points fits on a 35° friction angle. Whereas, the lab tests reveal a reduced strength for the 150 kPa test. Even though, the DEM results of the triaxial test differ partially from the lab tests, they fit very well to the theoretical assumptions.

Regarding the triaxial tests, it can be seen that the DEM is also able to model the strength softening after the reaching peak strength. A difficulty of the model is the creation of a certain void ratio, which has to be done during the insertion phase. The problem is that there will be always some settlements that will change the desired void ratio. Therefore, the secant stiffness modulus can differ from the lab results due to inaccuracies in the initial void ratio.

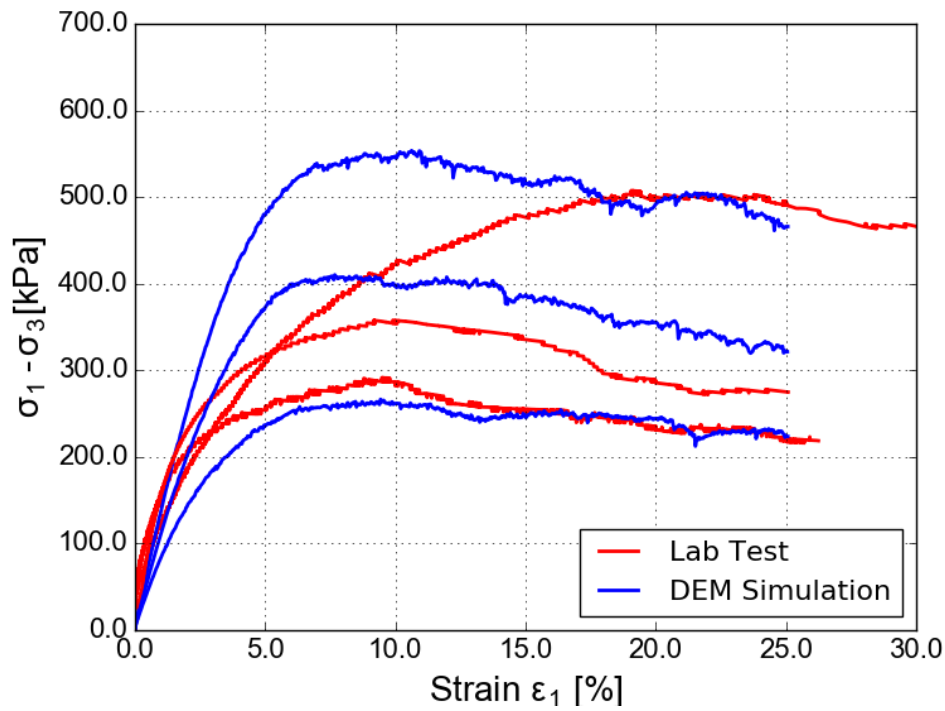


Figure 30: The stress strain curve of a triaxial test from lab experiments and from DEM simulations for a confining stress of 100, 150 and 200 kPa.

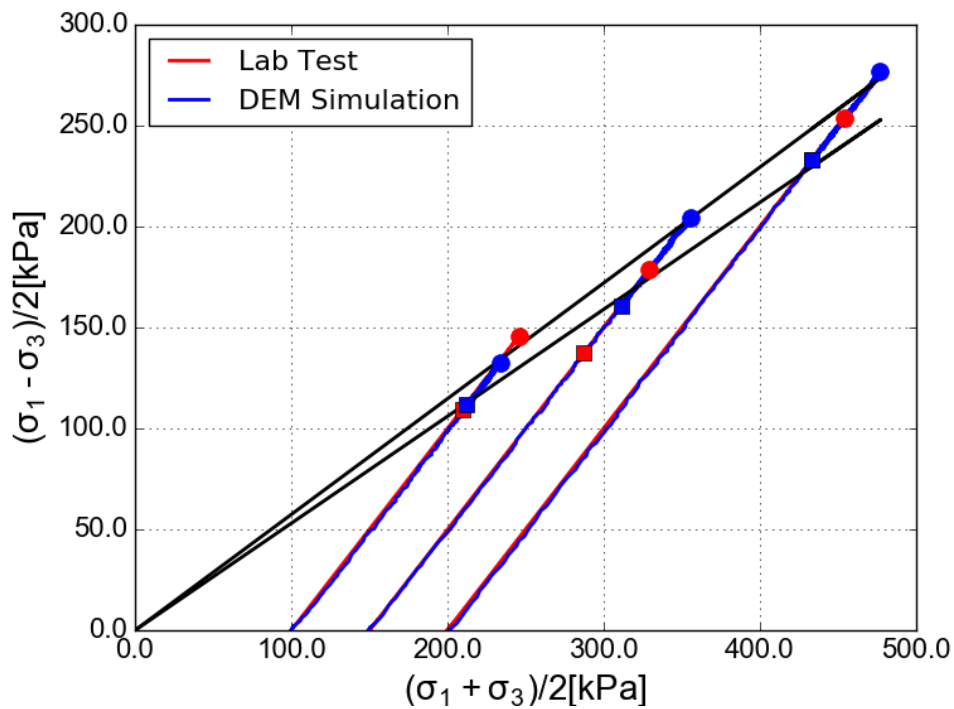


Figure 31: The stress path of a triaxial test from lab experiments and DEM simulations.

The results of the oedometer test show that the hardening behaviour of the sand can be well modelled, but the settlements after unloading are much smaller in the simulations than in the experiments. This behaviour is most probably caused by the elastic behaviour of the contact model, which is necessary for a stable simulation. In Figure 32 the stress strain is curve plotted for the oedometer test. The results from the DEM simulation are in good agreement with the lab results for the primary loading path until 320 kPa. The increasing stiffness of the soil due to compaction is obtained in the DEM simulation but the amount of compaction is less than in the lab test. Thus, the DEM code is not able to model the unloading behaviour of the soil with its original contact models.

This issue is negligible for the case of a constant driven probe, where no unloading appears. However, in the dynamic penetration process, an unloading is present after each stroke. For the dynamic penetration, the lower compactibility of the material will result in large rebounds after each stroke. Therefore, a lot of work was invested in developing new contact models, especially for the rolling resistance, see chapter 2.

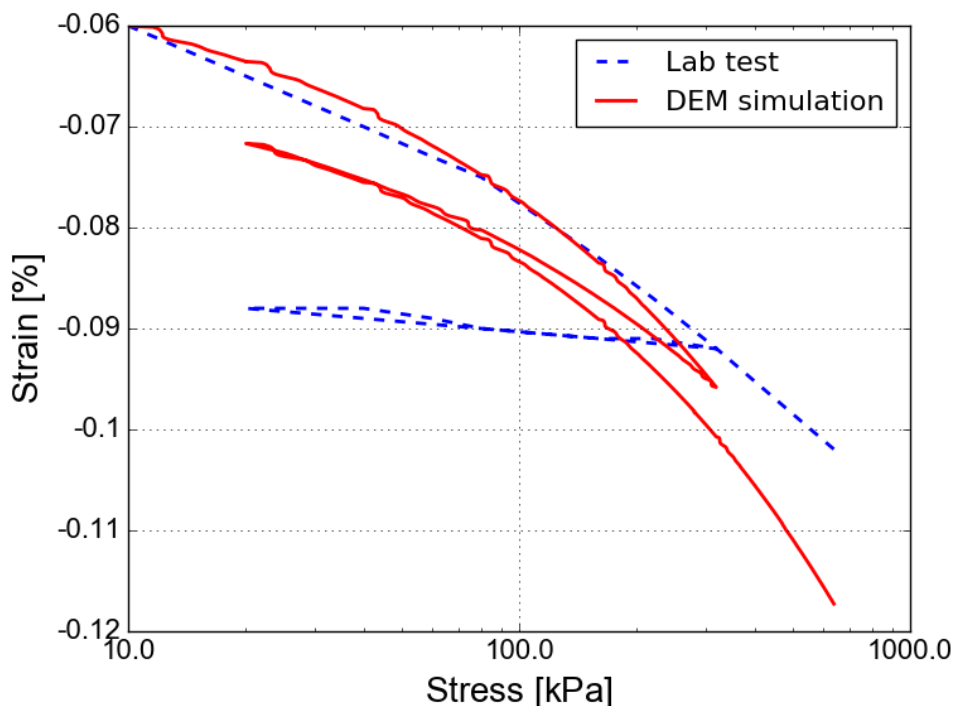


Figure 32: Stress strain relation from the oedometer test.

4 The Penetration Model

Three simulation models were developed to investigate the performance of HP³. A quasistatic cone penetration test was used to determine an approximate resistance and to study the influence of Martian gravity on the penetration resistance. A simplified model of a wall that penetrates with a prescribed velocity was used to determine the influence of an upscaled particle size and to investigate the soil behaviour under cyclic loading. The upscaling of the particle size is realised by a shift of the particle size distribution to larger particle sizes, where the relative distribution is unchanged. The final simulation is a fully coupled model that simulates the hammering mechanism as well as the soil response. For this purpose, the driving mechanism of HP³ needs to be modelled and coupled with a model of the probe inside the test bed.

The dynamic penetration process of HP³ is driven by the hammering mechanism consisting of a hammer mass and a suppressor mass, as well as connecting springs. This mechanism has to be implemented in the simulation, because the generated force profile of each stroke cycle is dependent on the soil response. Therefore, the masses of the casing, hammer and suppressor are modelled out of particles with a diameter of 5 cm and connected by the *fix spring* command of LIGGGHTS to represent the assembly in Figure 33.

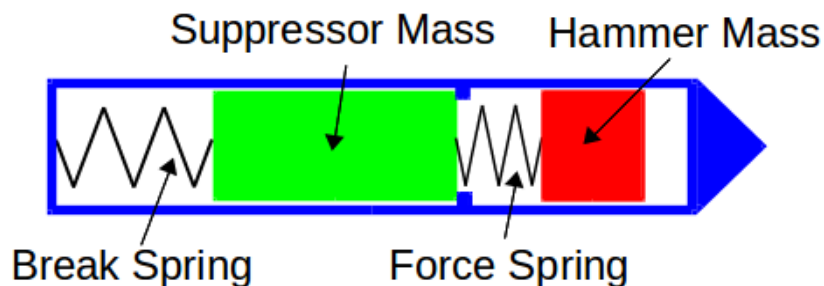


Figure 33: Hammer mechanism: Suppressor mass (green), Hammer mass (red)
Casing (blue)

The force spring that connects hammer and suppressor mass is much stiffer than the brake spring that connects the suppressor mass to the casing. The exact values are listed in Table 3.

Table 3: Mechanism parameters

Parameter	Description	Value for HP ³
m_{hammer}	Hammer mass	110 g
$m_{suppressor}$	Suppressor mass	460 g
m_{casing}	HP ³ casing mass	300 g
k_{force}	Stiffness of force spring	6222 N/m
k_{brake}	Stiffness of brake spring	73 N/m

In the initial position, the break spring is pre-compressed to a length of 52.35 mm, whereas its uncompressed length is 108 mm. The fully compressed length of the break spring is 29.35 mm and thus it has a spring deflection of 23 mm. The force spring gets fully compressed at each stroke cycle to a length of 20 mm, whereas the uncompressed length is 35 mm. The free flight distance of the hammer mass is 15 mm at the point of release.

In the DEM model, the complete mechanism is modelled out of 4 particles, where two particles represent the casing and the other two particles represent the hammer and the suppressor mass, see also Figure 34. The hammer and the suppressor mass are integrated only in z-direction by implementing a *fix nve/z* command. The two particles representing the casing are connected by the *fix rigid* command with an integration in the vertical z-axis. The velocity of the casing particles is computed and applied to a body of the penetrator that is inserted in the soil model. While the penetrator body is pushed forward through the soil, the resistive force on the penetrator is computed and applied back onto the particles that represent the casing mass. In this way, the simulations of the soil penetration and the hammering mechanism are coupled to simulate the dynamic penetration process of HP³.

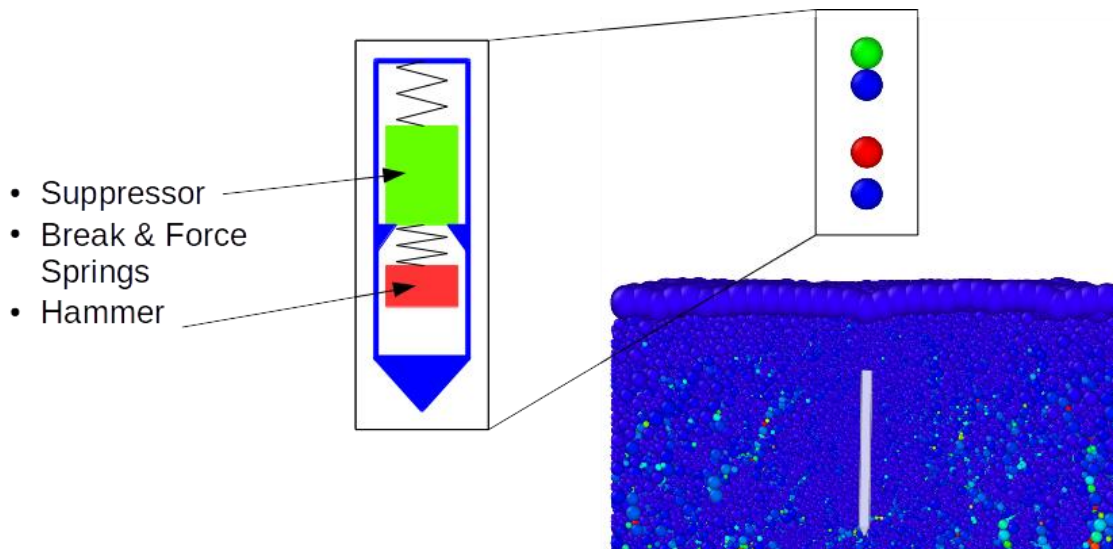


Figure 34: Implementation of the hammering mechanism in LIGGGHTS

The used DEM code allows for parallel processing, where the complete simulation domain will be divided in subdomains. Each subdomain is solved by a single processing unit. An overlap of the subdomains is necessary to compute the interaction between them. For the dynamic penetration model, the allocation of the domains for the central processing units (CPUs) should be selected with care. The penetrator body should always be in the same CPU domain as the particles of the hammering mechanism, otherwise errors may occur and lead to wrong results. That is why, this simulation should use an allocation of the CPU domains in horizontal direction with an uneven number of processor units for each direction. In this way, the centre of the domain is always computed by a single processor unit.

4.1 Simulation preparation

The HP³ penetration tests at DLR in Germany are performed in a cylindrical chamber with a radius of 40 cm and a height of 5 metres. In order to have a feasible representation, the simulation model has the same radius but is limited in the vertical height. The bottom in the simulation model is more than 30 cm away from the tip of the penetrator to avoid reflections or the creation of single force chains directly to the bottom. For the soil above the penetrator, the model uses a layer of particles atop of the probe to reproduce the overburden pressure in different depth. Therefore, the weight of the material above the investigated domain is calculated and assigned to the particles of the top layer. Furthermore, the soil model is subdivided in 4 domains in radial direction with larger particles in the outer domains to reduce the total amount of particles in the simulation. The smallest particles are used in the core of the simulated soil domain, where the interaction of the penetrator and the soil takes place.

In Figure 35 is a picture of the soil domain from side and bottom view, where the particles of each domain have a different colour. It can be seen that there is no large interpenetration of the subdomains, which is desired to avoid a mingling of the particle size distribution. Therefore, frictionless walls separate these subdomains in the beginning until the particles are settled. The particle scale of each subdomain is in the range of 1.2 to 1.5 times the neighbouring subdomain, whereas larger differences in the particle scale would cause too much interpenetration of the subdomains and a poor transfer of stresses at the subdomain boundaries.

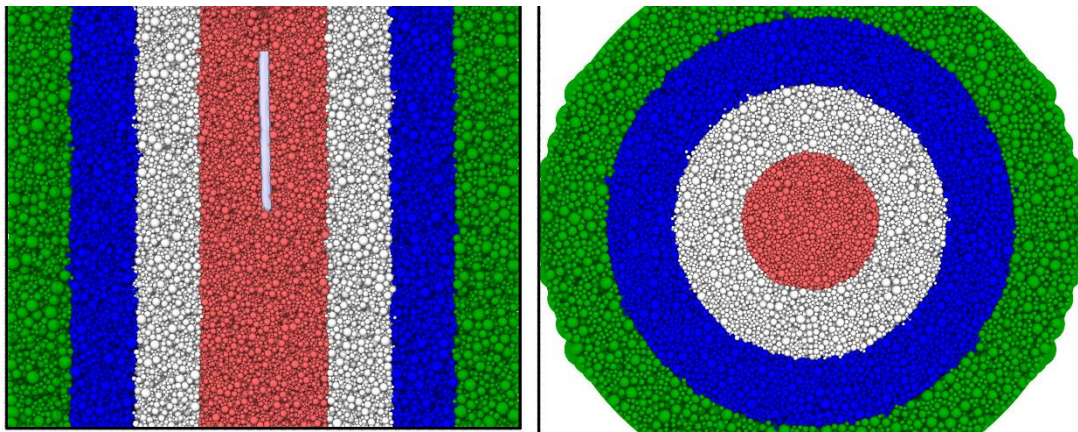


Figure 35: The subdivided soil domain with different coloured particles to highlight each subdomain (left: sliced side view, right: bottom view)

The skin size for the generation of the neighbor lists is set depending on the largest particle radius of the core domain. This reduces the amount of neighbouring particles and accelerates the computation, see chapter 2.7. The outermost cylindrical wall and the bottom wall have a friction value for sliding of 0.3 and for rolling of 0.1. The overburden particles have no friction at all and the penetrator has a coefficient of friction for sliding of 0.3, whereas no rolling resistance is applied. The rolling resistance between particles and the penetrator is ignored because of the small contact area between the grains and the smooth penetrator surface. The interparticle parameters are determined by the calibration procedure that is described in chapter 3.

4.2 Validation of particle size scaling for the penetration simulation

The usage of a scaled particle size instead of the real grain size is common for geotechnical applications in DEM (Falagush et al. (2015), Ciantia et al. (2016), Butlanska et al. (2014)). The upscaling of the particle size is used to reduce the total amount of particles and needs to be validated for each model on its own. In the simple element tests (e.g. triaxial shear test, oedometer test), where the stress distribution is homogeneous, the scaling of the particle size has less influence than

in a cone penetration test with local deviations of stress. In the case of simulations where the stress is locally induced into the soil, the upscaling should be used with caution.

In geotechnical research, it is known from experiments that particle size distribution has an influence on the penetration resistance of a cone penetration. This is mostly caused by the difference in the void ratio and stiffness, but less attributed to the particle size itself. Bolton et al. (1999) did investigations on the effects of different cone diameters on the penetration resistance in Leighton Buzzard sand. Different cone diameters B of 19.05, 10 and 6.35 mm were investigated to determine the grain size effect using the same material at same relative density. The results shown in Figure 36 reveal no particle size effect for tests on a sand (d_{50} : 0.225 mm) using cone diameter to mean grain size ratios of 85, 44 and 28. However, for medium sized particles (d_{50} : 0.4 mm) the ratios of cone diameter to grain size of 48 and 25 result in a similar resistance, whereas for the ratio of 16 the resistance is slightly higher but within acceptable limits. For the large particles (d_{50} : 0.9 mm), the influence of the grain size becomes more pronounced for the B/d_{50} ratio of 7, but is still negligible in comparison to the effect of void ratio on the penetration resistance.

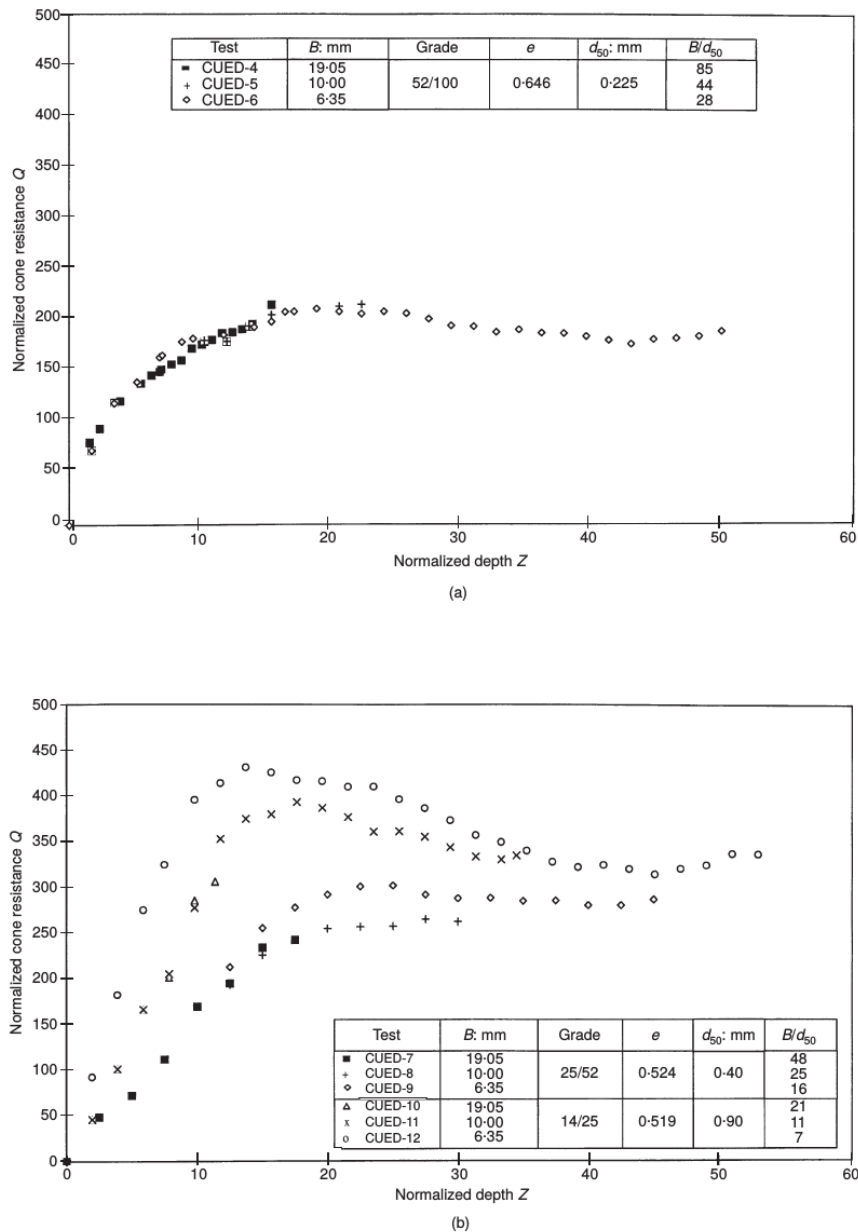


Figure 36: Grain size effects in Leighton Buzzard sand: (a) fine particles; (b) medium and coarse particles. Normalized cone resistance over Normalized depth (Bolton et al. 1999).

The increase of penetration resistance with smaller B/d_{50} ratio would cause an inaccuracy that decreases the penetration rate of a dynamic penetration simulation. Thus, the results are on the conservative side by underestimating the penetration rate of HP³.

Several simulations of cone penetration tests with constant velocity are evaluated to investigate the effect of particle size scaling in the DEM. These simulations differ from the experiments of Bolton et al. (1999) in so far as the particle scale is increased and the penetrator diameter is kept constant. Thus, the larger particles have more weight than the small scale particles to obtain the same density and the same stress state. The stiffness of the particles is also changed at upscaling due to

the radius dependent stiffness of the Hertz model. Thus, the same stiffness behaviour in the macroscale is obtained.

Two different models were used for the investigations of particles size effects on the penetration resistance. A model using large particle size scales in the range of 20 to 50 and a model for small particle size scales in the range of 5 to 20 are investigated. The penetration rate for the simulations is 1 cm/s for the small scale model and 10 cm/s for the large scale model, where the penetration rate in a dry cohesionless sand has a minor influence on the penetration resistance until the particles inertia gain impact on the resistance. This was also identified from the experiments by Bolton et al. (1999), where cone penetration tests with penetration rates of 2.5 mm/s and 20 mm/s were performed in a dense dry specimen without noticeable deviations in the resistance.

The particles for the overburden pressure are not applied for the quasistatic cone penetration models. The bulk density after gravity loading is about 1450 kg/m³ with a grain density of 2720 kg/m³. The particles' Young's modulus is 0.8 GPa and the Young's modulus of the penetrator and the confining walls is 21 GPa. The Poisson's ratio is set to 0.3 and the coefficient of restitution for damping is 0.1, where no distinction between particles and walls is made.

The penetration simulations using a large scaling of the particle size are modelled in a test chamber with 40 cm diameter and a depth of 75 cm. The total penetration depth for these simulations is 55 cm, where the penetrator tip dips into the material after 5 cm. Thus, the effective penetration depth is about 50 cm. The chamber is radially subdivided in cylindrical domains with larger particle scale outwards (see Figure 35), as explained already in chapter 4.1. The interparticle friction parameters are 0.6 for sliding and 0.4 for rotation. The particle-wall friction coefficient for sliding is 0.3 and for rotation is 0.1, whereas the particle-penetrator friction coefficient for sliding is 0.3 and for rotation is 0.0. The rolling resistance between particles and penetrator is negligible because of the small contact area between the grains and the curved shape of the penetrator casing. The investigated scale sizes are 50, 30, 25 and 20 with penetrator diameter to mean grain size B/d_{50} ratios of 1.2, 2, 2.4 and 3, respectively.

The particle displacement profiles in Figure 37 show the areas of displaced particles. The limited colour range may be exceeded, wherefore the red coloured particles are displaced 5 mm or even more. It can be seen that for the smaller particle size, the area of displaced particles is smaller, whereas the penetration resistance is not directly related to this. The penetration resistance shown in Figure 38 reveal a converging of the resistive force with a particle scale of 30 and lower. Small deviations are attributed to irregularities of the soil structure, as it is in real soils. The scattering of the actual resistive force becomes less with smaller particles due to a more homogeneous stress distribution. The differences in resistance from a particle scale of 30 and 20 is below 10 %. Thus, a penetration

simulation using a particle size that is scaled by the factor of 30 yields already well approximated results with respect to the penetration resistance. The deformations in the soil due to the penetration has a more extensive effect for the coarse-grained simulation, when comparing the particles displacements of particle scale 20 and 30.

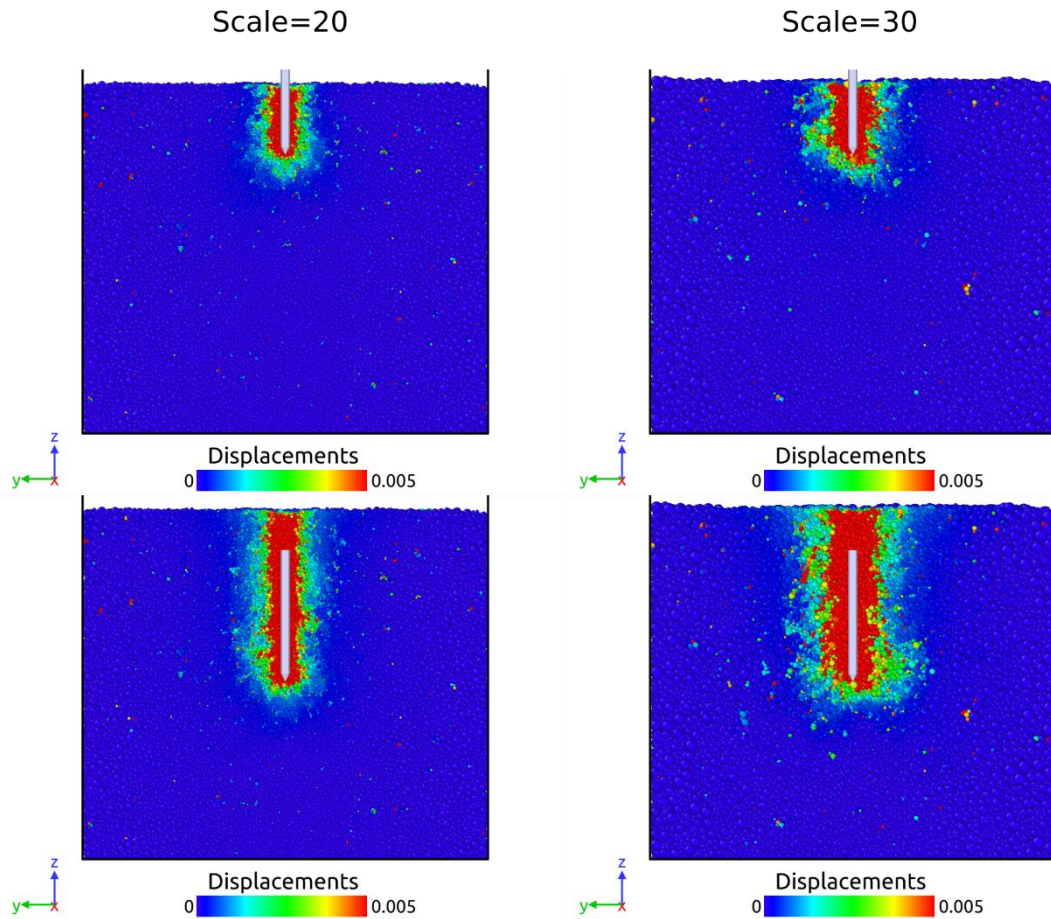


Figure 37: Comparison of particles displacements in m for different particle scales after 20 cm and 40 cm of penetration

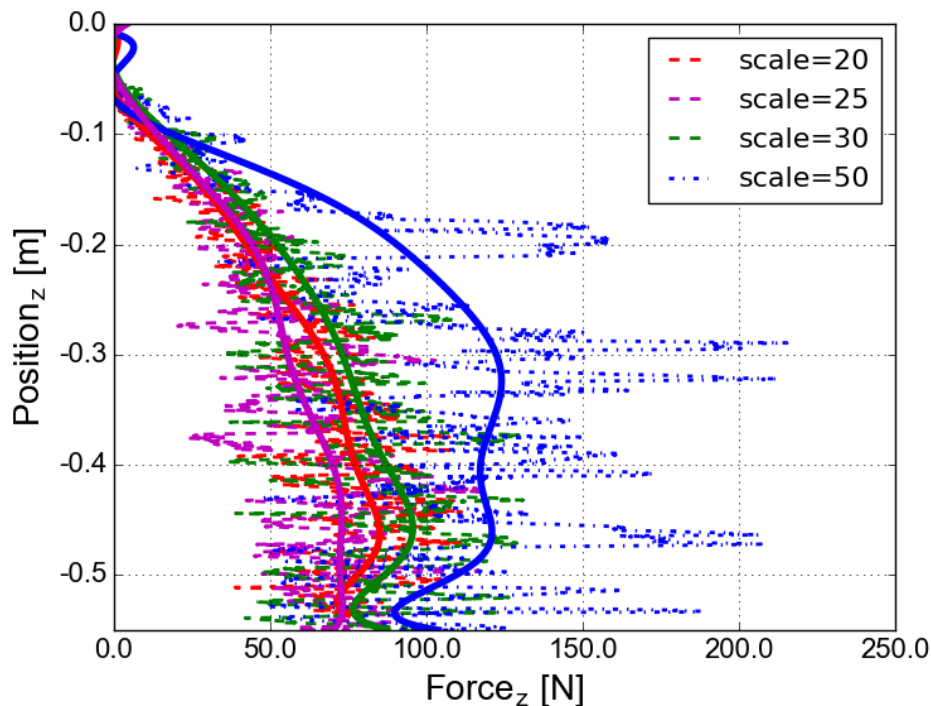


Figure 38: Comparison of the penetration resistance using different scale sizes.

The simulation of the whole test chamber with a scale of particle size lower than 20 would take too much computation time. Therefore, a small scale model of a wall that penetrates into a small soil specimen is used to investigate the influence of the particle scale on the penetration resistance and the soil behaviour. The shape of the wall from a side view corresponds to the shape of the HP³ penetrator. The investigated particle scales are 20, 15, 10 and 5 which correspond to B/d_{50} ratios of 3, 4, 6 and 12. For the small scale model the total penetration depth is 5 cm. The particle parameters are similar to those from the large scale model.

The comparison of the areas of mobilised particles using different scales of the particles size is given in Figure 39. It can be seen that the areas of similar particle displacements are almost the same, where the exact shape of these areas becomes clearer with smaller particle scale. This means that in all simulations the same amount of particle volume needs to be pushed in a similar way to penetrate in. A similar rupture plane and a good agreement in the resistive force with different particle scales validate the coarse-grained model.

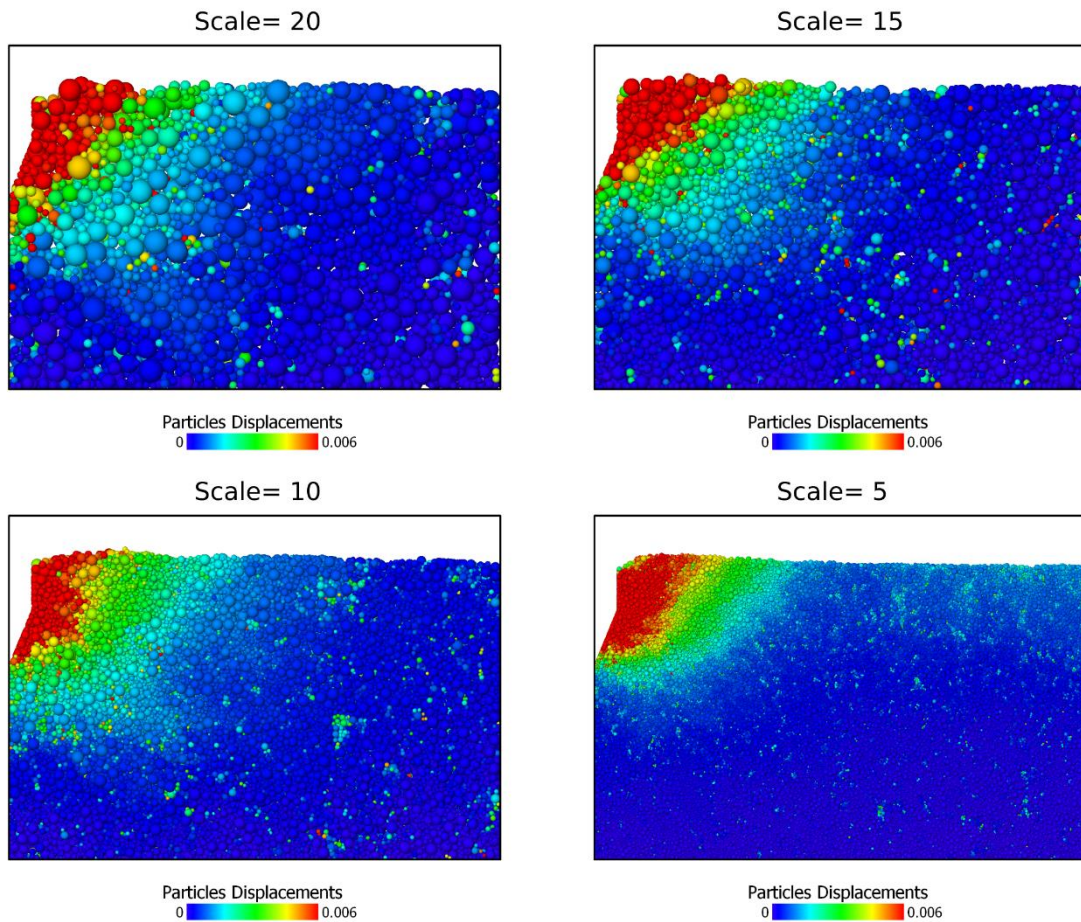


Figure 39: Comparison of the particles displacements in m after 5 cm penetration using different particle scales.

The penetration resistance for all simulations of different particle scales are displayed in Figure 40. The simulations using larger particle scales reveal a larger scattering in the resistive force, but the mean value corresponds well in all simulations. At the beginning of the penetration, the resistive force of the larger scaled models increases a bit faster due to the larger particles that need to be mobilised right from the beginning.

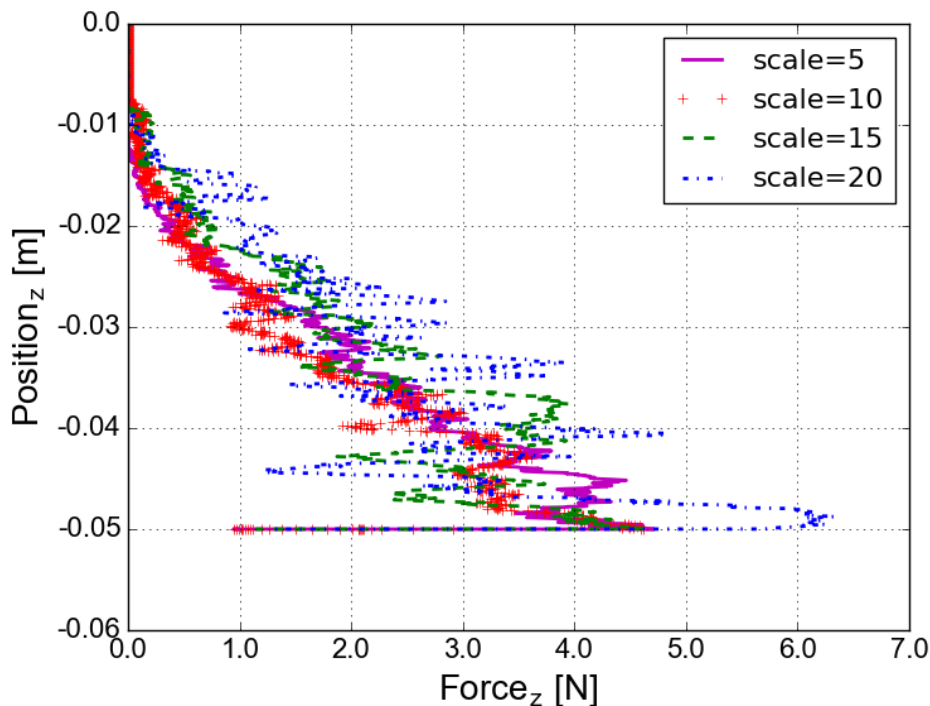


Figure 40: Comparison of the resistive force during the penetration of the wall at different particle scales

4.3 Simulation of quasistatic cone penetration tests

The quasistatic cone penetration test is an ideal model to evaluate the influence of the soil parameters such as relative density, particle friction and gravity on the penetration resistance. The procedure of a cone penetration test is already explained in the previous chapters. Briefly summarised, a probe penetrates into the sand under a constant velocity, while the resistive force is measured. The penetration rate for the simulations is 10 cm/s in vertical direction, unless otherwise specified. The penetrator drives centrally into the cylindrical soil specimen. The interface friction between the penetrator and the soil is 0.3 for sliding, whereas no rolling resistance is assumed in the interface due to the smooth surface of the penetrator. Different gravitational constants, soil parameters and densities are investigated and the influence on the resistance is analysed. There is no overburden pressure applied for the quasistatic penetration. The particles' radii are scaled by a factor of 25.

The influence of the penetration rate on the resistance is shown in Figure 41. The density in this simulations is about 1485 kg/m^3 and corresponds to a void ratio of 0.83, which is a loose packing. The tangential friction parameter is 0.6 and the coefficient of rolling resistance is 0.3. It can be seen that the penetration resistance is little affected by penetration rates less than 1.6 m/s for a penetration depth up to 0.3 m. At a depth of 0.3 m, the asymmetric stress distribution results in a varying

resistance. The penetrator is only moved vertically and cannot compensate a non-uniform stress distribution, see Figure 42. If an additional integration of the penetrator's motion in the horizontal plane is considered, it could compensate the unilateral stress and thus result in a more unique resistance.

At a fast insertion of 4 m/s the resistance increases more significant right from the beginning. At such fast insertion rates, the inertia force of the particles and probably the damping in the normal contact begin to have an impact on the resistance. Bolton et al. (1999) found by experiments in dry sand that penetration rates of 0.25 cm/s and 2 cm/s yield almost the same resistance, where faster insertion rates were not studied. In the simulation model, much higher penetration rates are investigated. The negligible effect of the penetration rate holds only for dry sand, whereas in saturated sand the fast insertion can produce excess pore water pressure depending on the permeability of the material.

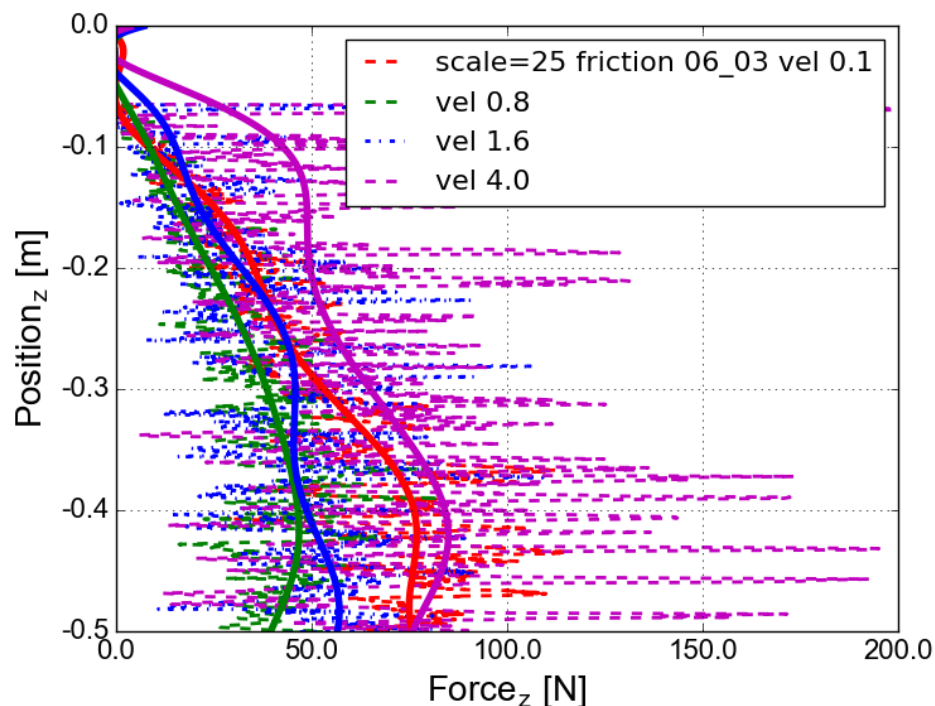


Figure 41: Comparison of penetration resistances at different insertion rates for quasistatic penetration.

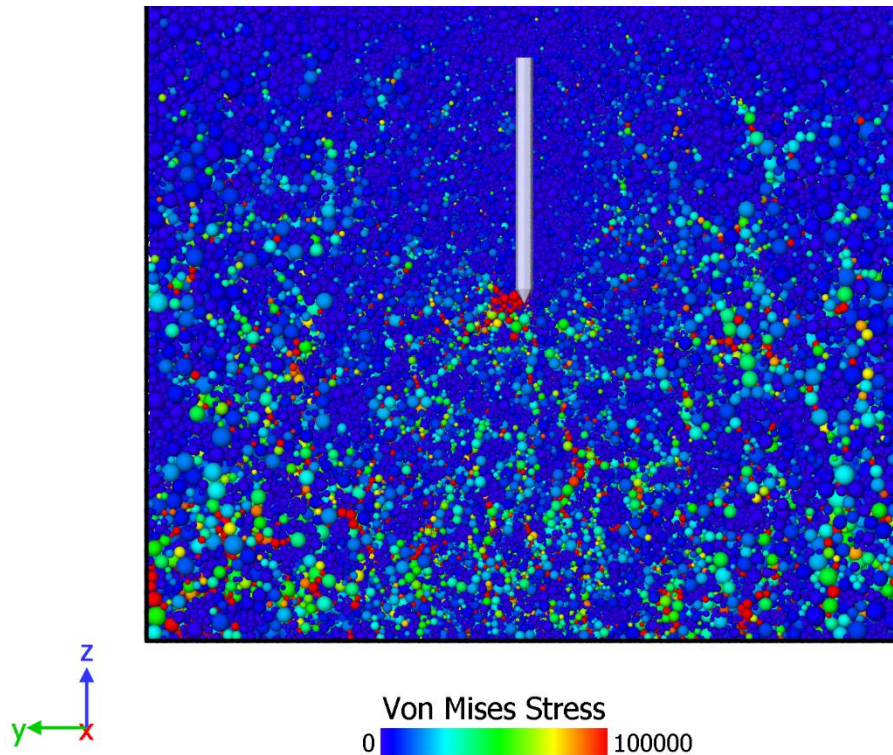


Figure 42: Von Mises Stress in Pa. The influence of fabric anisotropy on the stress distribution, if the penetrator is not integrated in x- and y-direction.

The comparison of the DEM simulations to laboratory results is plotted in Figure 43. The lab results of the cone penetration are taken from Lichtenheldt et al. (2014). They carried out several penetration tests in the 5 m testbed at DLR and determined a mean value and a confidence interval of the measurements. In difference to the DEM simulations, the results from the lab tests are solely the tip resistance without the shaft friction. Therefore, the frictional force is subtracted from the total resistance. Two DEM simulations for Earth gravity and a bulk density of 1430 kg/m^3 with a void ratio of 0.9 were done applying different interparticle friction parameters. The *Friction0604_density1.43* denoted data had a coefficient of friction of 0.6 and a rolling resistance value of 0.4. The *Friction0705_density1.43* denoted data had a coefficient of friction of 0.7 and a rolling resistance value of 0.5 instead. An additional simulation under Martian gravity with a friction coefficient of 0.6 and a rolling resistance of 0.4 is denoted by *marsgrav*, where a density of 1453 kg/m^3 was achieved. The simulation using the lower friction values in Earth gravity fits almost into the confidence interval of the measurements, where it has to be considered that without the shaft friction, which is about 10 % of the total resistance, the data would fit even better. A simulation with a perfectly smooth penetrator surface would not only ignore the shaft friction, but also affect the tip resistance. Therefore, the shaft or the tip resistance cannot be identified in particular. The simulation using the larger friction parameters is completely outside of the confidence interval. From the penetration simulation in Martian gravity it can be obtained that the penetration resistance is decreased significantly

due to the lower stress level in the soil. This of course has an impact on the penetration performance of HP³ and needs to be considered.

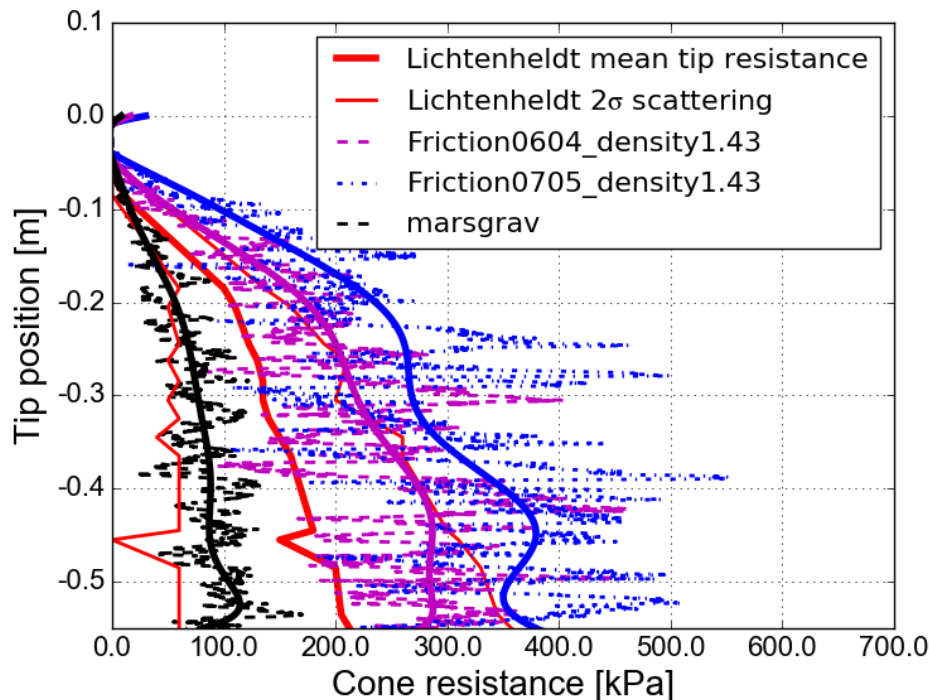


Figure 43: The cone resistance in kPa from DEM simulations compared to the tip resistance of cone penetration test in the 5 m testbed at DLR from Lichtenheldt et al. (2014).

4.4 Simulation of dynamic penetration of HP³

The heat flow and physical properties probe (HP³) of the NASA InSight Mission is able to penetrate itself into a granular material by an implemented hammering mechanism. This driving mechanism is modelled separately above the soil specimen in the DEM simulations. The generated driving force depends on the soil response, wherefore the simulation of the hammering mechanism is directly coupled with the penetrator model in the soil specimen. The setup of the simulation can be seen in Figure 44. The coupling between the driving mechanism and the penetrator is done by an exchange of velocity and force. Therefore, the velocity of the blue coloured particles for the casing is computed and applied to the penetrator in the soil, while the force that acts on the penetrator body is applied back to the blue particles. Furthermore, rotations of the penetrator are permitted to avoid artificial bracing of the penetrator in the soil. It was observed from simulations with a locked rotation of the penetrator that directional force chains arise at the tip and the back end and generate a torque. These forces would usually cause a rotation of the penetrator, whereas for a rotational locking these forces increase the horizontal stress and thus the shaft friction is increased.

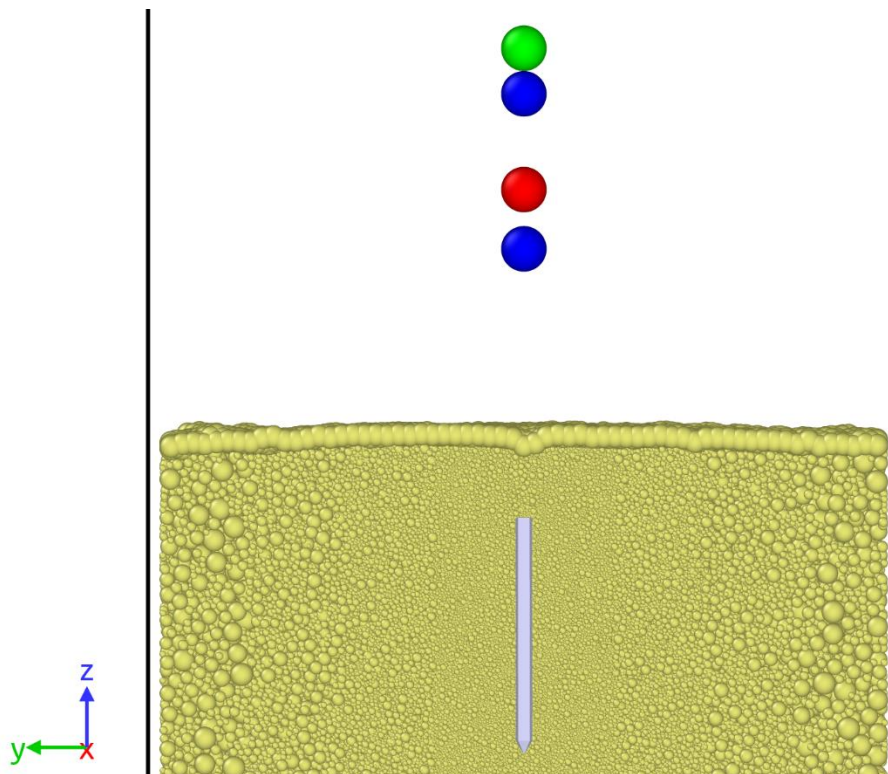


Figure 44: The model of the hammering mechanism above the soil domain. The hammer mass in red, the suppressor mass in green and the casing in blue.

The domains of different particle scales are obvious, where no intermixing of particles between these domains occur, see also chapter 4.1. The smallest particle scale is used in the centre, where the interaction of penetrator and soil takes place. The overburden pressure is applied by the particle layer that lies on the soil. The penetrator is pushed into the material with a velocity of 0.5 m/s to obtain the initial setup for the hammering action. As soon as all oscillations disappeared, the simulation of the hammer strokes begins. Therefore, the corresponding springs are applied to the particles and accelerate them.

The penetration rate per stroke cycle is about a few millimetres, which is quite small in comparison to the particles' size. The standard contact models for friction and rolling resistance behave highly elastic for small deformations, which leads to incorrect results at unloading after each penetration stroke. Hence, it was necessary to develop a new contact model that allows for plastic deformation even for small displacements. An improved contact model for tangential friction and rolling resistance was developed within this work and is explained in more detail in chapter 2.5. The main feature of the improved contact model is the plastic displacement of particles right from the beginning of the shear or roll motion. The model could not yet be used for the simulations due to a discontinuity in the moment. The problem of the discontinuity appears only in the three-dimensional case and lead to an unstable behaviour of the particles. Therefore, only slightly modified versions for sliding friction and rolling resistance were applied. Those

models have a limited damping and the current resistive value results from the total deformation and is not based on the resistance of the previous time step.

The DEM model of the HP³ penetration provides information on the penetration resistance and the performance of the driving mechanism. The penetration resistance depends on the void ratio of the soil, the interparticle friction parameters, the stress level and the penetration rate. Whereas the performance of the driving mechanism depends on the resistance against the penetration and the shaft friction at rebound. The shaft friction is necessary to prevent the probe from a backwards motion.

It was observed from the simulations that depending on the penetration rate different soil failure mechanisms appear. Either the penetrator begins to open a cavity by pushing the material sideways or the penetration strokes compact the soil beneath the penetrator, causing a punching in of the penetrator. The different penetration types can be identified by the penetration resistance profile. In the case of a cavity opening, the penetration occurs under a more or less constant resistant force, whereas a linear increasing resistant force with penetration indicates a compaction in front of the penetrator. The results of a simulation of 3 dynamic stroke cycles are shown in Figure 45. Each stroke cycle consists of a major stroke by the hammer mass at the beginning, followed by some minor strokes of the hammer mass and a stroke of the suppressor mass. The first strokes by the hammer mass are marked in the plots. The rebound after the strokes varies between the cycles, where in the third stroke cycle it can be seen that after the large rebound of the first stroke, the second stroke of the suppressor mass penetrates even more. This is possibly caused by a loosening of the material due to the large unloading. From the force displacement profile at the bottom of Figure 45 different penetration types can be observed. The third stroke increases linearly from 40 N up to a resistance of 110 N and indicates thus a structural compaction owing to this stroke. The second stroke instead penetrates at a more or less constant resistance of 60 N, which indicates a cavity opening in front of the tip.

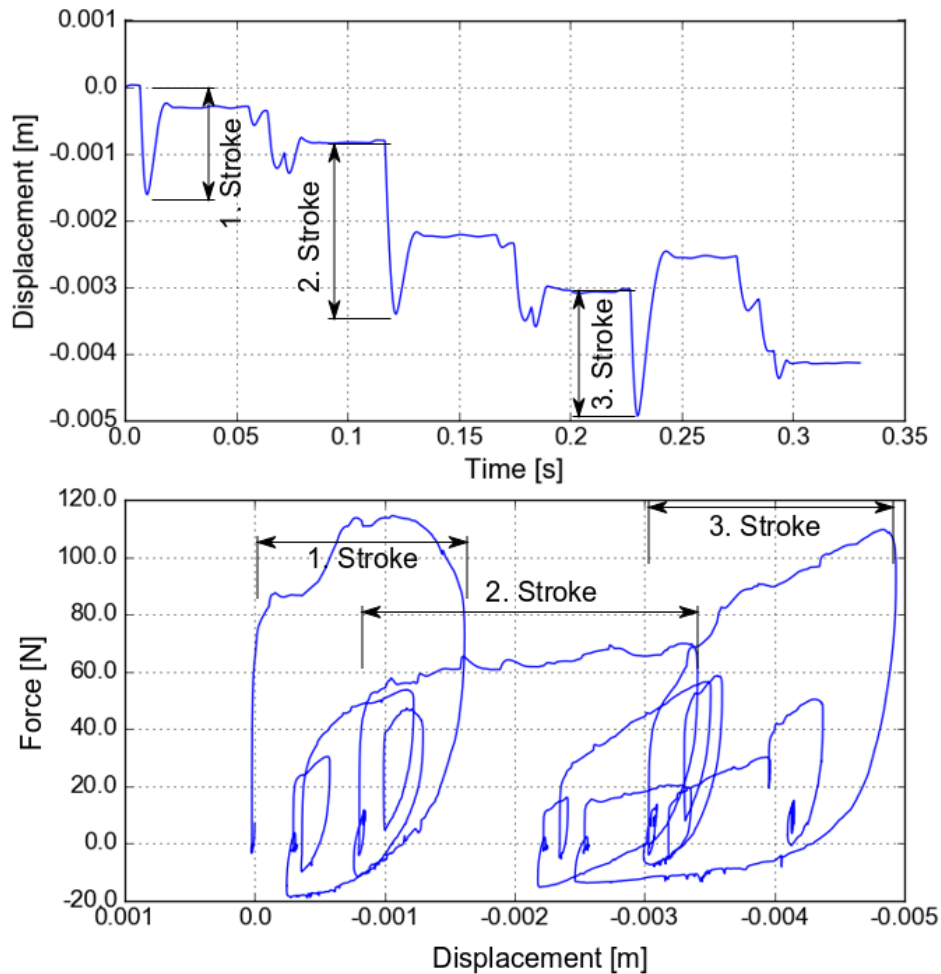


Figure 45: Simulation results of 3 stroke cycles. The tip displacement at the top and the force displacement curve on the bottom.

A snapshot of the particles' displacements at the point of maximum penetration during the first hammer stroke is shown in Figure 46. The red coloured particles may exceed the displacement of 0.1 mm. It is obtained that the displacements in the soil propagate more horizontally than vertically over time. The largest displacements appear in the direction of the surface normal of the tip. It would be preferred to have less impact on the soil structure in front of the tip to avoid an increase of the tip resistance by compacted soil in this area. The induced stress by the first hammer stroke is illustrated in Figure 47 and Figure 48 for the vertical and the horizontal stress. For this purpose, a snapshot of the corresponding stress was taken before the hammer stroke appeared and is subtracted from the stress at the time step where the hammer stroke is acting. Figure 47 illustrates the induced vertical stress at the time step just after the impact of the hammer mass. It can be seen that the induced stress spreads from the tip conically into the ground. The vertical stress at the shaft is reduced instead. A similar plot of the horizontal induced stress at the time of the first hammer stroke is given in Figure 48. The

horizontal stress propagates laterally from the tip, whereas a reduction of the horizontal stress appears below the tip and at the cone shoulders. A validation of the horizontal stress distribution at the moment of the stroke was done with an axisymmetric FEM model of the penetration in PLAXIS, see Figure 49. In the FEM simulation a prescribed displacement of 1 mm was applied to the penetrator and the dynamic response of the soil was investigated. For the interface, a strength reduction factor of 0.8 was used. The used soil model was the Hardening Soil Model with small strain stiffness. The friction angle is 31° and the E_{50}^{ref} stiffness is 10 MPa, whereas the un-/reloading stiffness is 30 MPa. A dry density of 1500 kg/m^3 with a void ratio of 0.7 was used. All other parameters were chosen based on experience. The values in Figure 49 range from a tensile stress of 2 kPa in blue to a compressive stress of -35 kPa in red. White coloured zones exceeded the colour range. The stress distribution from the FEM simulation confirms the results from the DEM simulation. A reduced horizontal stress in front of the tip and at the cone shoulders is obvious. The increased horizontal stress lateral from the cone tip is also present in the FEM simulation.

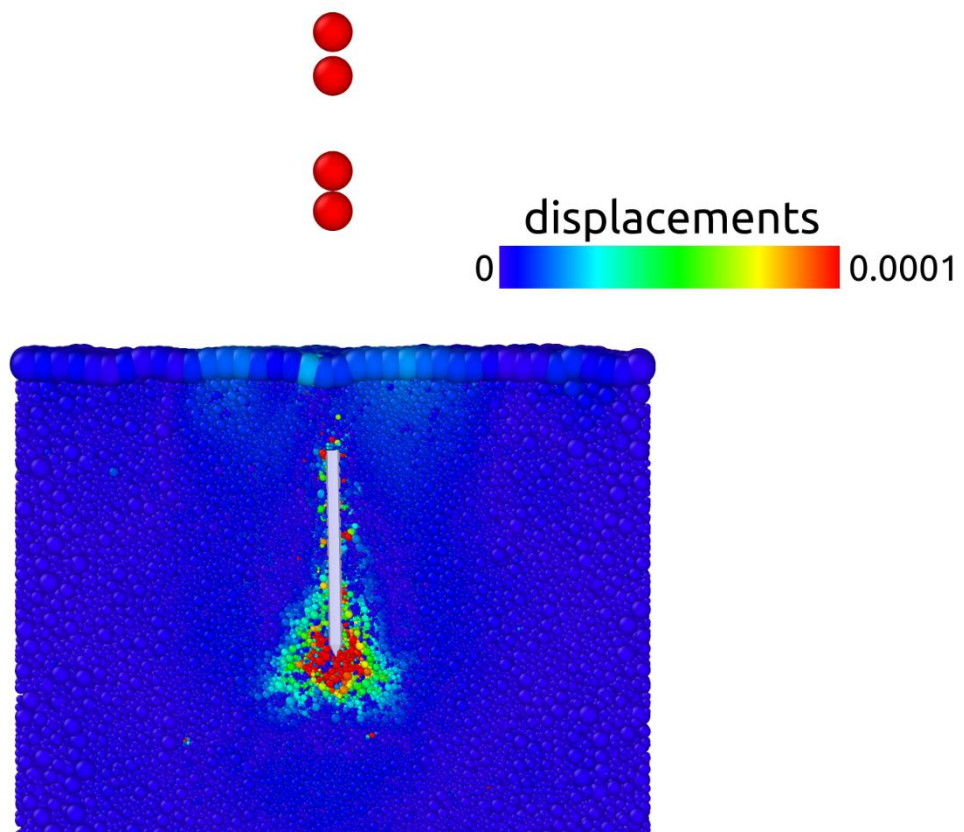


Figure 46: The particles' displacements in m due to the first stroke of the hammer mass.

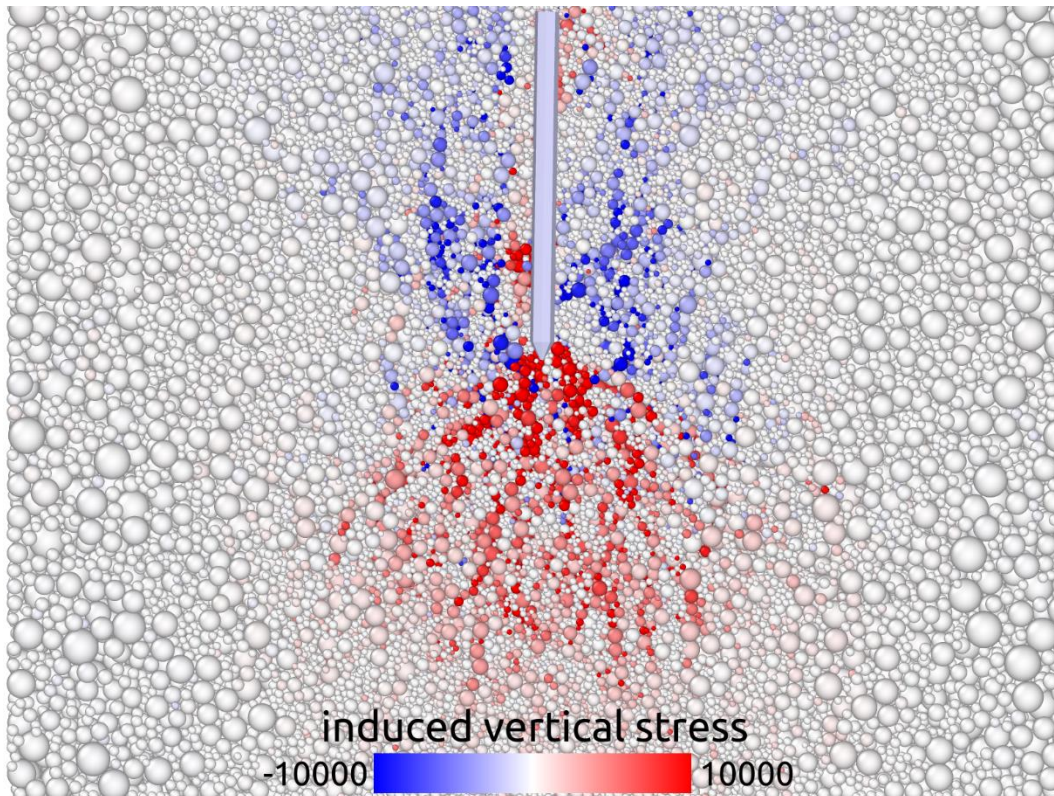


Figure 47: The induced vertical stress in Pa at the time of the first hammer stroke. The maximum values exceed the colour range.

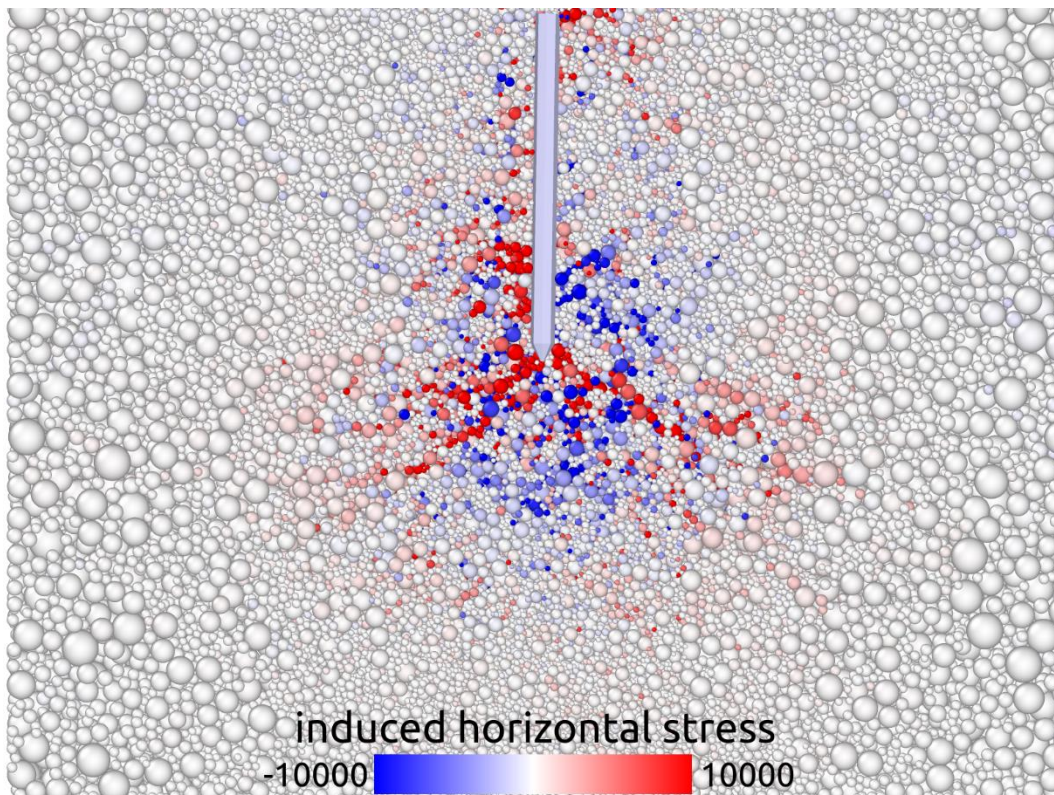


Figure 48: The induced horizontal stress in Pa at the time of the first hammer stroke. The maximum values exceed the colour range.

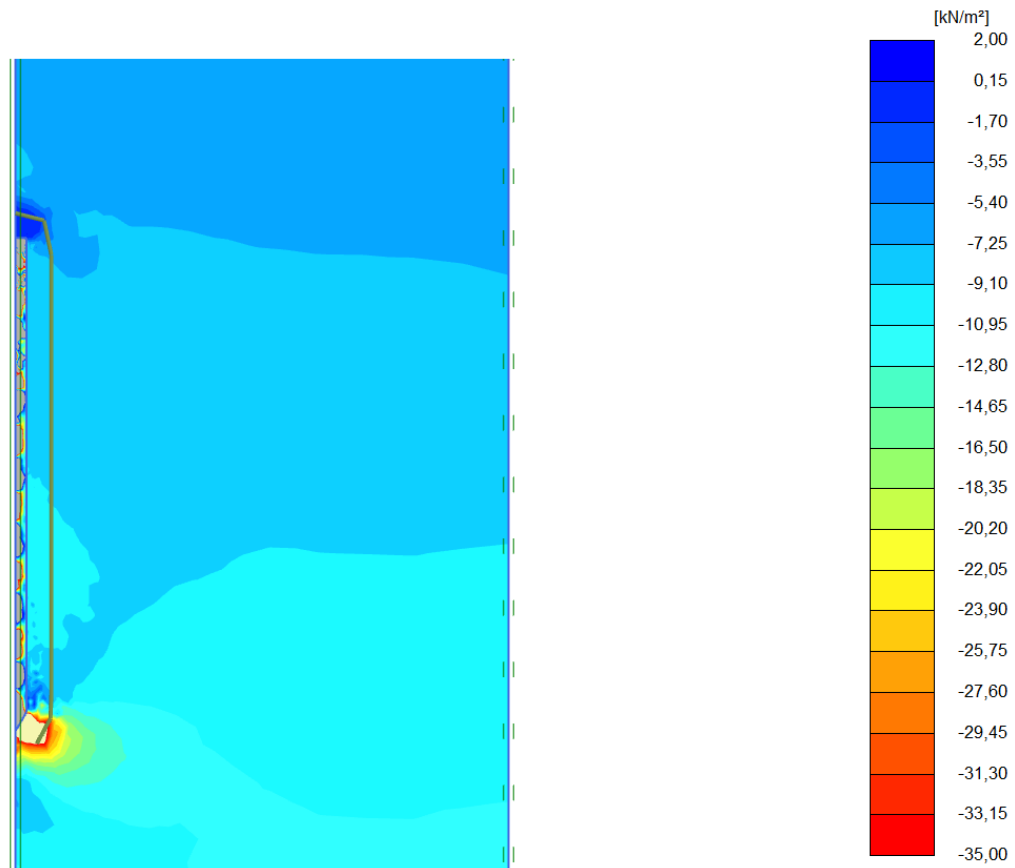


Figure 49: Effective radial stress in kPa from a FEM simulation in PLAXIS.

The dynamic penetration was also investigated in different depth. For this purpose, the overburden pressure was adjusted to produce a stress level in the soil that corresponds to different depth. From the resistant force during the first hammer stroke in Figure 50 an increase of the resistance with depth is obvious. The resistance ranges from 100 N to about 220 N for 1 m of overburden to 4 m of overburden. Furthermore, an increase of the shaft friction during the rebound of the probe is observed. The shaft friction at rebound ranges from about 10 N to almost 50 N.

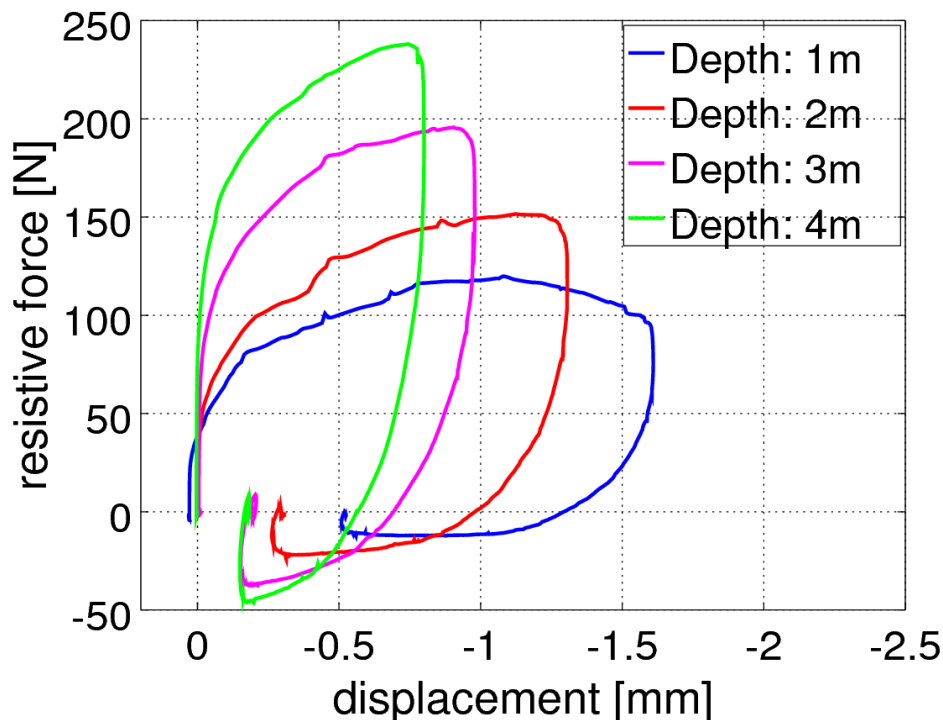


Figure 50: Resistant force due to the first hammer stroke in different depth.

For the validation of the dynamic penetration simulation, the known penetration rates from lab tests at DLR and other simulations of the HP³ penetration were used for comparison. Similar simulations of the HP³ penetration were performed by Lichtenheldt et al. (2016) and revealed displacements of about 2 mm per stroke cycle in a depth of 1 m. The corresponding results in Figure 51 show the displacement profile from a DEM simulation compared to results from a one dimensional multi body simulation (MBS) and measurements from lab tests. In comparison, the second stroke cycle from the simulation results in Figure 45 yielded a similar penetration of about 2 mm. In contrast, the first and the third stroke cycle penetrate only about 1 mm during a full loading cycle. The large rebounds after each stroke indicates that the simulation of the dynamic penetration has some inaccuracies that can be traced back to the elasticity in the contact models. All the elastic behaviour of the particle structure is a result of the elasticities in the contact models, since no other elastic behaviour is defined within the code.

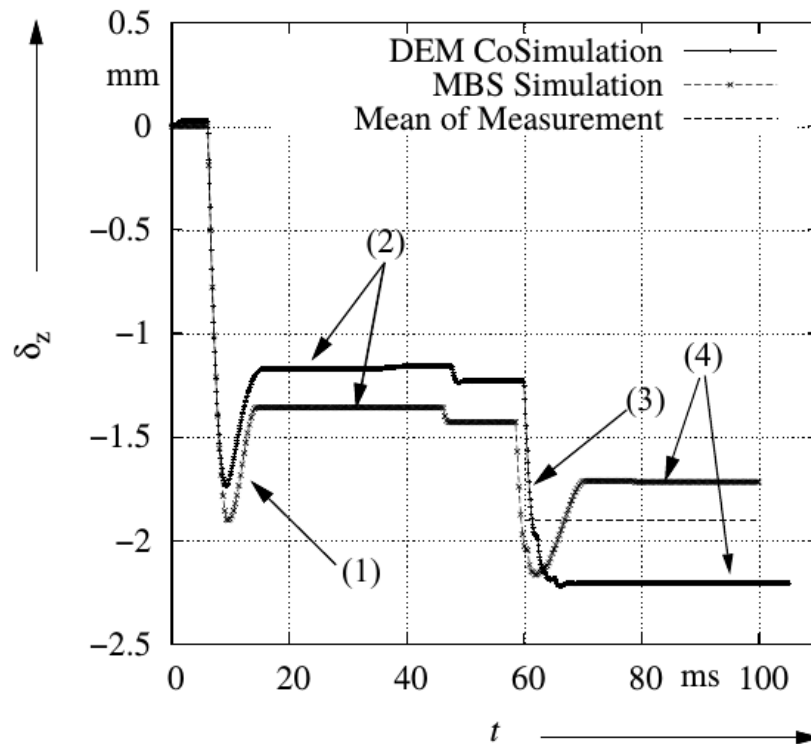


Figure 51: Displacements of HP³ during a stroke cycle from a DEM simulation, a 1D multi body simulation (MBS) and from measurements of lab tests. The penetration was performed in 1 m depth. Lichtenheldt et al. (2016)

The penetration rate per stroke cycle from the lab tests is determined as a mean value of a few penetration cycles. Owing to the fact that the penetration rate is determined by optical measurements of points on the trailing cable, the precision is too inaccurate to obtain the real penetration curve of each stroke cycle. The penetration rate from a full penetration of about 6000 stroke cycles is given in Figure 52. With the fact that a stroke cycle appears every 3.6 s, a mean value for a single stroke cycle can be determined. In the beginning of the measurements, HP³ penetrates with a rate of about 2 mm per cycle, whereas in the depth between 3 to 4 m the penetration rate decays. At a depth of 4 m, the penetration per cycle is reduced to a value of about 0.13 mm. The reason for the decay of the penetration performance is not clarified yet. It is known from lab tests in the 3 m sample that the penetration is not affected by approaching the bottom. Therefore, the tip resistance or the shaft friction must have changed somehow. A certain shaft friction is necessary for the hammering mechanism of HP³ to absorb the rear-facing force of the suppressor mass. Thus, a reduction of the shaft friction may reduce the penetration rate. The reduction of the shaft friction in dynamic installed piles can appear and is known as friction fatigue. The phenomenon of friction fatigue is explained in more detail in chapter 5.5.

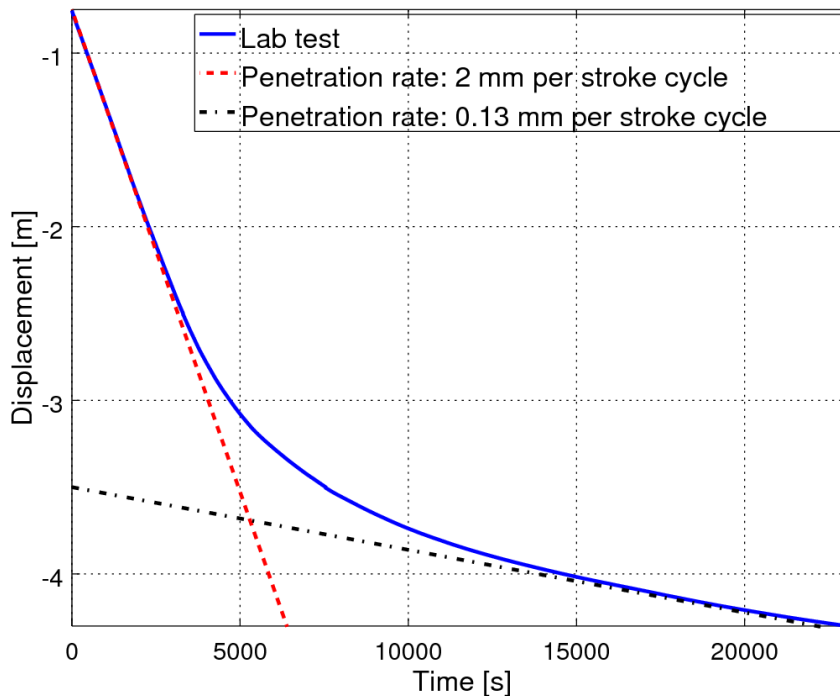


Figure 52: Measured penetration rate of HP³ plotted over time from DLR laboratories and penetration curves of constant penetration rates. A stroke cycle appears every 3.6 s.

4.4.1 Plane strain model with prescribed displacements

The plane strain model of a wall penetration is used to obtain the influence of the hammering action onto the soil deformation. Therefore, a wall with a profile of the probe penetrates into a 20 cm wide, 4 cm thick and 12 cm high particle domain. Periodic boundaries are used in the x-direction with a domain thickness of 4 cm. The periodic boundary inserts the particles that leave the domain at the opposite boundary. Thus, the horizontal stress in x-direction is the same at the boundaries and a plane strain condition is produced. The wall penetrates into the domain with a prescribed velocity. Therefore, the velocity profile of a stroke cycle was determined first by a dynamic simulation of the HP³ penetration. The velocity profile and the corresponding displacement of the probe can be seen in Figure 53. The movement of the wall in the horizontal direction is locked.

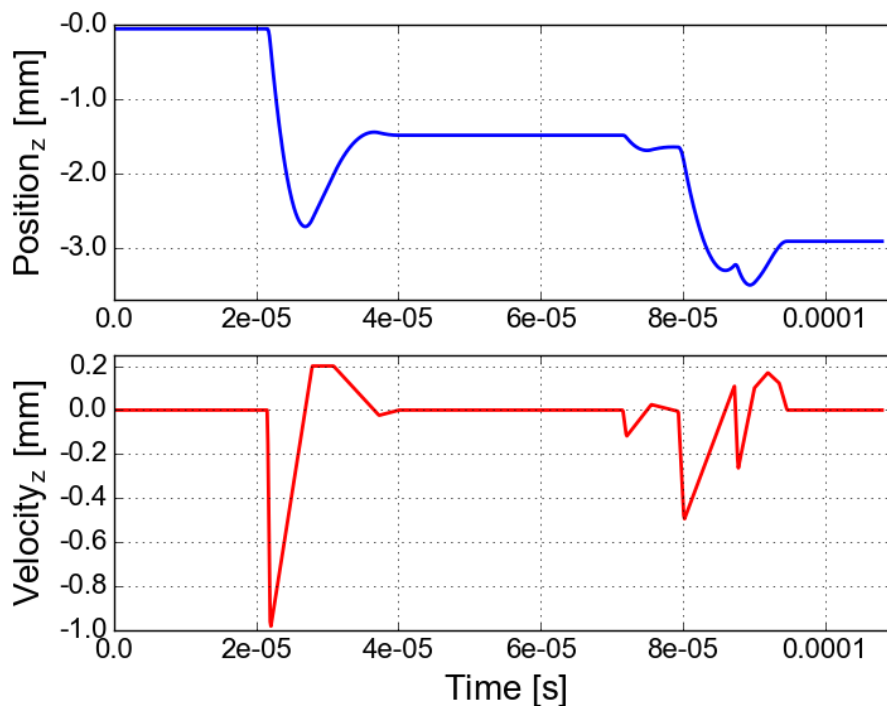


Figure 53: The vertical position of the penetrator from a 3D simulation of a hammer cycle at the top and the corresponding velocity at the bottom.

For the preparation of the initial condition, the wall penetrates the first 5 cm with a constant velocity of 1 cm/s into the soil domain. After the insertion is done, the velocity profile of 10 dynamic stroke cycles is applied. The dynamic penetration phase causes about 3 cm of settlements. The soil deformations due to the dynamic cycles is observed.

The plane strain model allows to reduce the soil volume and thus the particles' size can be reduced. The model is no representation of the HP³ penetration but the soil deformations are comparable. Furthermore, relative comparisons between quasistatic and dynamic penetration can be drawn.

The particles displacements in Figure 54 reveal that the particles are dragged downwards with the penetrator.

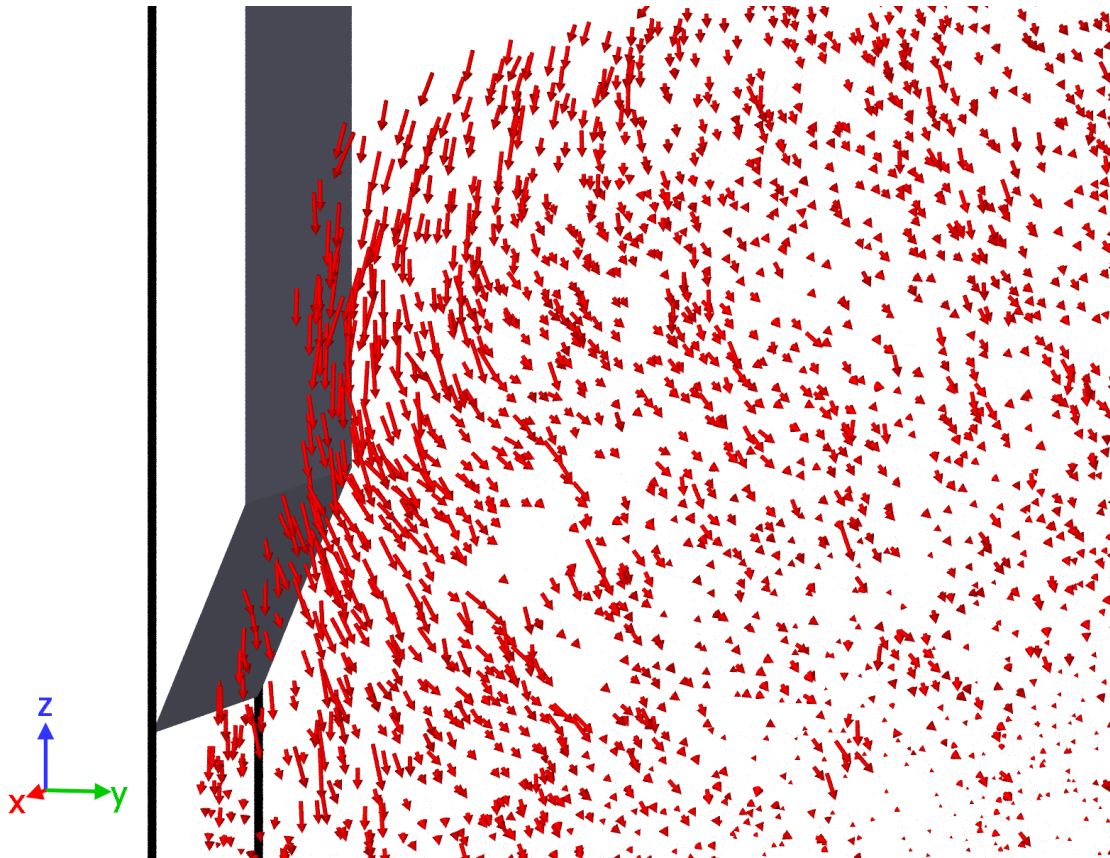


Figure 54: The particles displacements after a few stroke cycles.

The comparison of the influence of a dynamic and a quasistatic penetration is given in Figure 55. Therefore, a model of a quasistatic penetration with a rate of 1 cm/s and a total penetration depth of 8 cm is used as a reference. In the end of the quasistatic penetration, the penetrator is kept in its final position for a time to get rid of time dependent effects.

At the top of Figure 55, a checked pattern out of different coloured particles is used to obtain the volumetric deformations from 3 cm of penetration. Therefore, the particles are coloured with two different colours in the shape of a regular mesh at 5 cm of penetration. The mesh gets distorted due to further penetration up to a depth of 8 cm. The lower level of the free surface and the more compacted mesh at the tip and the cone shoulder reveal a higher compaction of the soil due to the dynamic penetration than for the quasistatic one.

The corresponding displacements of the particles can be seen in the centre of Figure 55. It shows up that the particles near the penetrator are displaced more in the case of the dynamic than in the quasistatic penetration. This displacements are restricted to particles close to the penetrator. The particles in far distance from the penetrator are displaced to the same extend for the dynamic and the quasistatic penetration. This is also an indication for a higher compaction due to the dynamic penetration. The dynamic oscillation of the penetrator leads to a drop of particles

from above, which means that the particles are not just pushed sideways but also dragged downwards.

The Von Mises stress of the particles is shown at the bottom of Figure 55. The plots show the stress due to the initial filling phase and the complete penetration. It is obtained that the cyclic motion of the dynamic penetration reduces the stress near the penetrator. In the case of the quasistatic penetration, horizontal force chains are produced and clamp the penetrator. The reduced stress at the dynamic penetration can be feasible only with a concurrent compaction of the soil. This result confirms the assumption of the higher soil compaction at the dynamic penetration. The reduced horizontal stress results in a reduction of the shaft friction whereby the axial bearing capacity gets reduced due to the dynamic penetration. This was also found by investigations of the Deep Foundation Institute DFI (2015) on the bearing capacity of vibratory and impact driven piles. The focus of this investigations was on axially loaded piles, where the vibrated piles had about 80 % of the capacity of impact driven piles.

The total resistance for the dynamic and constant driven wall is plotted in Figure 56. The maximum resistance of the dynamic driven wall exceeds the resistance of the constant driven wall only slightly, if the first stroke at 0.05 m depth is neglected. The resistance during the second stroke by the suppressor mass is below the resistance of the constant driven probe. This is possibly caused due to the unloading of the first stroke that loosens the soil.

The main difference in the dynamic driven case is that after each hammer stroke the wall moves a bit upwards and relieves the soil. As a result, the stress in front of the penetrator tip is reduced. Furthermore, due to the cyclic motion of the wall in the dynamic case, the material gets more sheared and thus more compacted near the wall.

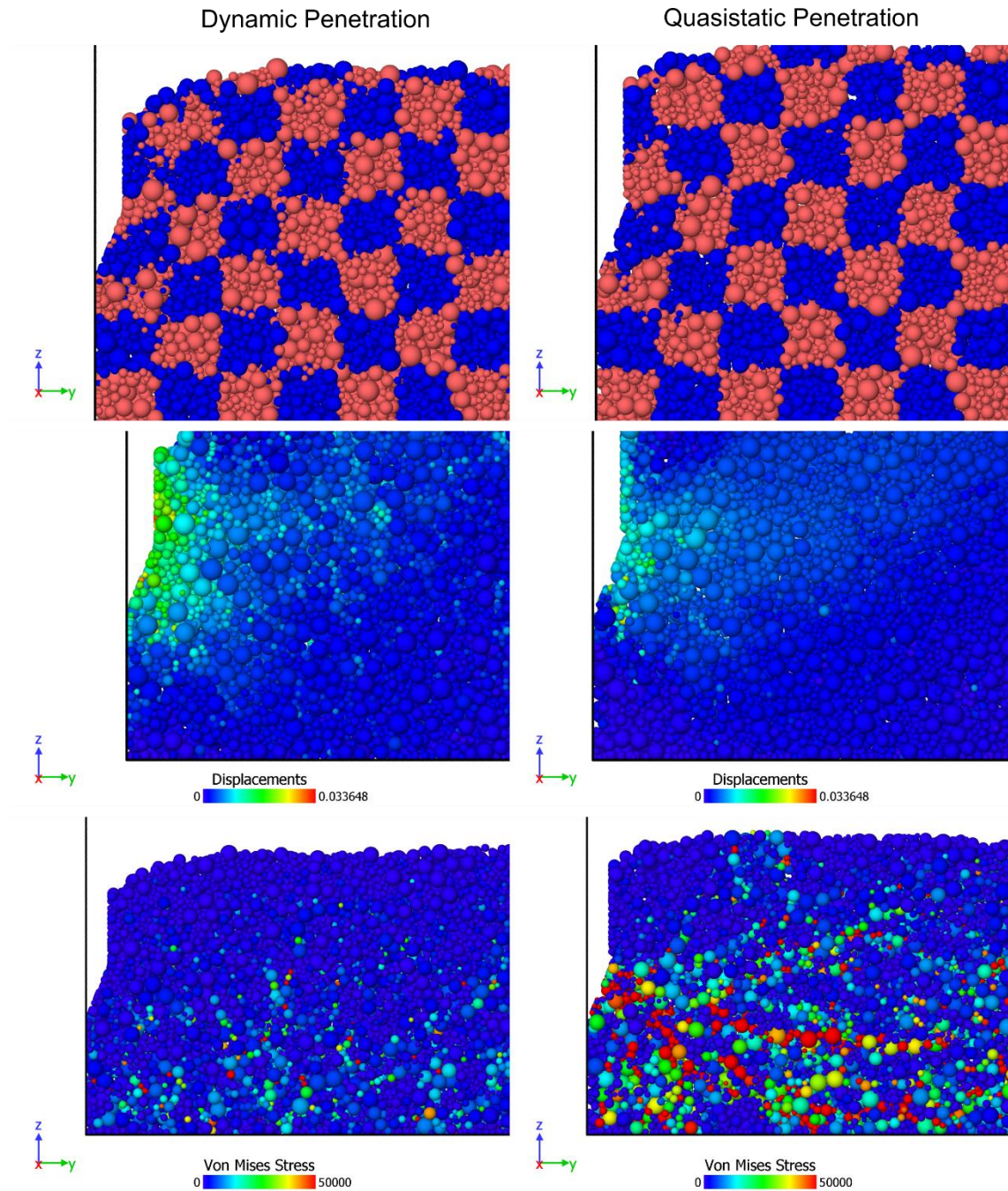


Figure 55: The deformed soil domain at the top, the particles displacements in m at the centre and the Von Mises stress of the particles in Pa at the bottom.

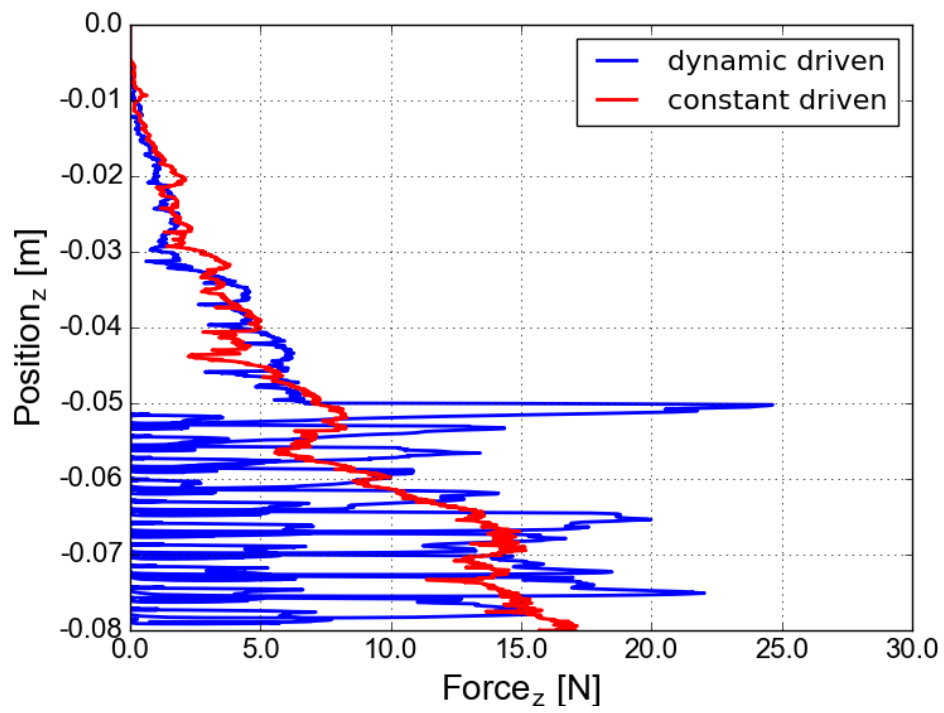


Figure 56: The resistance of the dynamic and constant driven wall. The dynamic penetration begins at 0.05 m depth.

5 A Pile Drive Model implemented in Matlab

The pile drive model in Matlab is developed to determine the influence of different tip and shaft resistance on the performance of the hammering mechanism. Therefore, the hammering mechanism is modelled by a multi-body system using spring-dashpot and collision contacts. The outer casing of the penetrator is subdivided in many elements to apply different shaft resistances over the length of the penetrator. In Figure 57 is a sketch of the penetrator model, where the collision contacts are defined between the hammer mass m_{13} and the tip mass m_{12} as well as between the suppressor mass m_{14} and an element of the penetrator's shaft m_8 . The shaft elements, the tip and the rear end are connected by stiff springs representing the material stiffness of the penetrator. The soil model is a spring-dashpot connection between the penetrator elements and a fixed point, where the spring characteristics allows irreversible displacements as soon as a maximum resistance is reached.

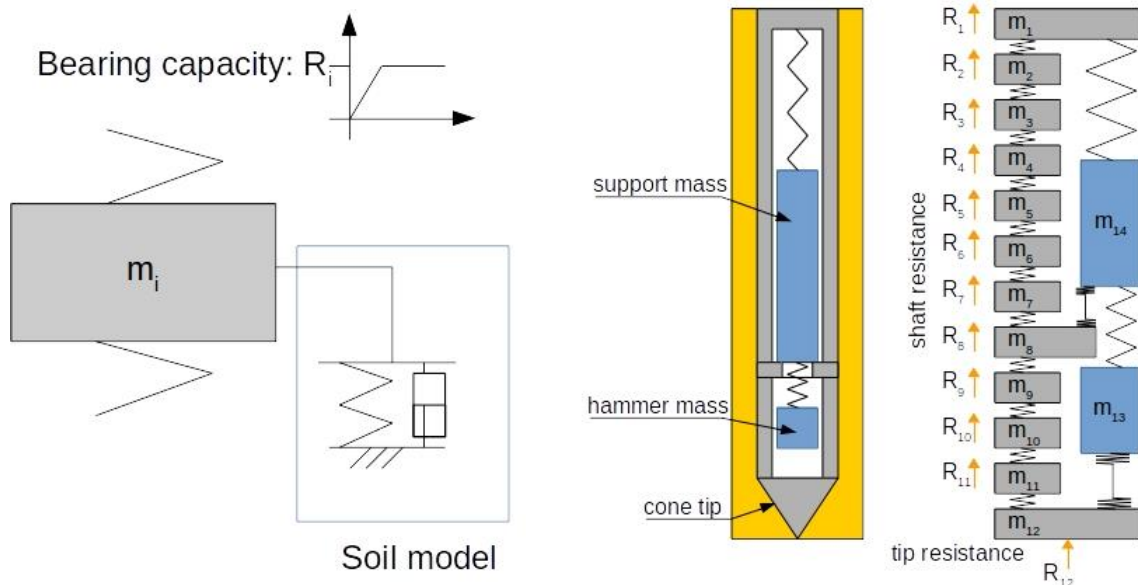


Figure 57: Sketch of the soil and the penetrator model

The pile drive model is based on the work of Smith (1962). The difficulty for this way of simulation is the determination of the resistive force acting on the tip and the shaft of the penetrator during the hammering strokes. The determination of an approximated tip resistance has been presented in Poganski et al. (2016) and will be explained in the next chapter.

5.1 An analytical approach for the penetration resistance

The penetration resistance can be divided in a tip resistance and a shaft friction. Both are acting against the penetration of the probe. The approach that is used for the estimation of a penetration resistance is the bearing capacity of foundations.

The tip resistance can be calculated based on Terzaghi's (1943) theory for a bearing capacity of a flat circular shallow foundation, where this approach is not valid for the deep penetration. Terzaghi's equation is usually restricted to a maximum footing depth of 3 times the diameter of the foundation. This is due to the fact that in deeper foundation the complete ground heave will not occur. Instead, the soil will compact locally to accommodate the displaced material. Nevertheless, Terzaghi's equation was used as a first approach for the computation of a tip resistance and a shaft friction. The assumptions that are made for the bearing capacity by Terzaghi are:

- Dry soil
- No inclination of the penetrator
- Flat tip
- Vertical penetration force
- Horizontal surface
- Horizontally layered soil
- Infinite half space

Under these restrictions the equation for the bearing capacity at the tip of the penetrator is:

$$R_{tip} = A(\gamma_u 2rN_\gamma + \gamma_o tN_q + cN_c) \quad (27)$$

with the radius r and the cross-section A of the penetrator, the soil specific weight underneath the tip γ_u , the soil specific weight above the tip γ_o , the current depth t and the cohesion c of the soil. The bearing capacity factors N_γ , N_q and N_c can be determined considering the preceding assumptions as:

$$N_{q0} = \frac{1 + \sin \varphi}{1 - \sin \varphi} e^{\pi \tan \varphi} \quad (28)$$

$$N_\gamma = (N_{q0} - 1)s_\gamma \tan \varphi \quad (29)$$

$$N_q = N_{q0} s_q \quad (30)$$

$$N_c = \cot \varphi (N_{q0} - 1) s_c \quad (31)$$

, where φ is the friction angle of the soil and s_γ , s_q and s_c are shape factors of the foundation. The shape factors for a circular foundation and a centric origin of the force are given by:

$$s_\gamma = 0.7 \quad (32)$$

$$s_q = 1 + \sin \varphi \quad (33)$$

$$s_c = \frac{s_q N_{q0} - 1}{N_{q0} - 1} \quad (34)$$

The additional resistance caused by the shaft friction between soil and penetrator can be added to the bearing capacity at the tip to derive the entire bearing capacity acting on the penetrator.

The shaft friction can be computed using the Coulomb's law of friction by integrating the horizontal acting stress over the shaft area and multiply it with the coefficient of friction between the penetrator and the surrounding soil. Therefore, the horizontal stress needs to be known. A first assumption is that the horizontal stress is about the half of the vertical stress. This is based on the approximation for normally consolidated soils, where the ratio of horizontal stress σ_h to vertical stress σ_v can be determined by

$$\frac{\sigma_h}{\sigma_v} = 1 - \sin(\varphi) \quad (35)$$

with the internal friction angle φ . A value of 0.5 corresponds to a friction angle of 30° . The vertical acting stress can be determined by the soil load at the depth t

$$\sigma_v = \rho g t \quad (36)$$

with the soil bulk density ρ and the gravitational constant g .

This approximation is not valid anymore during the dynamic penetration, due to the induced changes in stress by the penetrator, but it is valid for an undisturbed material at the beginning of the penetration process.

The resistance due to shaft friction R_{shaft} is given by the integration of the horizontal stress over the shaft area A_{shaft} times the interface friction coefficient μ_{inter} .

$$R_{shaft} = \int_{A_{shaft}} \mu_{inter} \sigma_h(t) dA \quad (37)$$

Since the shaft area is subdivided in many elements, the integration can be approximated by the sum over all shaft segments:

$$R_{shaft} = \sum_{i=1}^{n_{segments}} \mu_{inter} A_{segment,i} \sigma_h(t_i) \quad (38)$$

The determination of the real shaft friction force becomes difficult since the horizontal stress that acts on the penetrator is influenced by the dynamic load cycles. Experiments on monotonic and cyclic driven piles by Vogelsang et al. (2017) reveal a friction force at the shaft lower than the prediction from equation 37 supposing a medium dense packing with a dry density of 1440 kg/m³ and an interface friction coefficient between pile and soil of 0.3.

The penetration resistance for the pile drive model in Matlab increases linearly up to a defined maximum resistance at which plastic deformation occurs. The difference from this approach of a resistance force to a more sophisticated DEM model is shown in Figure 58. The more realistic resistance from the 3D DEM model increases slightly during penetration, where elastic and plastic deformation are produced at the same time. However, in the Matlab model elastic and plastic deformation are separated in time. The elastic part of the resistance in the Matlab model is necessary to achieve a stable simulation.

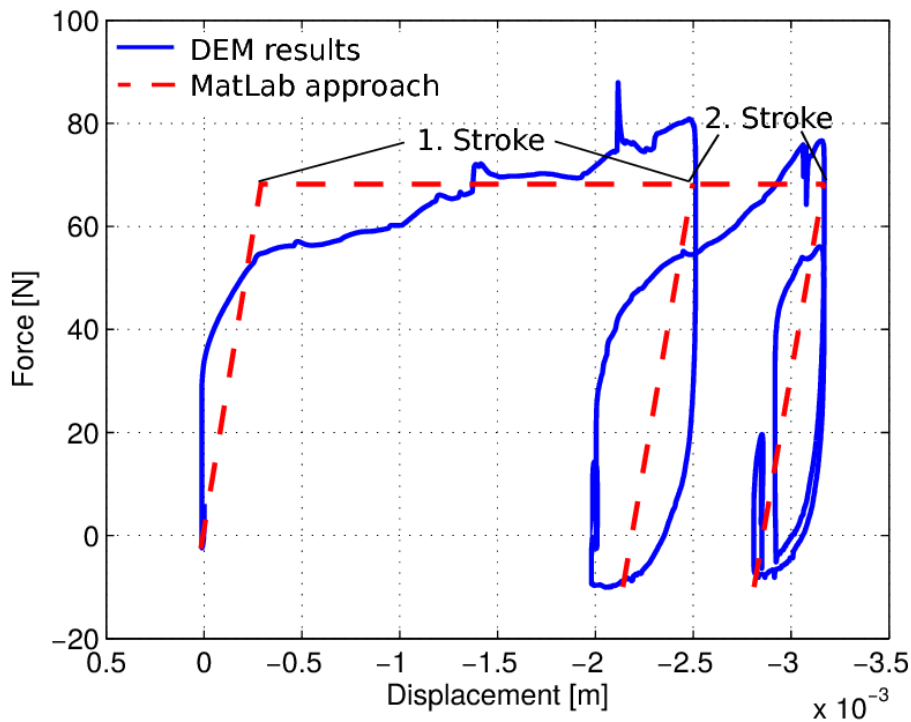


Figure 58: Force-Displacement curve over a stroke cycle. (Poganski et al. 2016)

From DEM simulation it was also observed that the resistance at the second stroke can be less than from the first stroke. This behaviour is not captured yet in the Matlab model, so that only a constant maximum resistance can be applied.

5.2 Time Integration

The time domain is integrated by an explicit Euler method:

$$\ddot{x}_k = f(x_k, \dot{x}_k) \quad (39)$$

$$\ddot{x}_{k+1} = \dot{x}_k + \ddot{x}_k \Delta T \quad (40)$$

$$x_{k+1} = x_k + \dot{x}_{k+1} \Delta T, \quad (41)$$

where x_k denotes the displacement at the calculation step k and ΔT is the time increment. The time increment is defined by the highest eigenfrequency of the system and is 10 % of the critical time increment ΔT_{crit} :

$$\Delta T_{crit} = \frac{2\pi}{\omega} = 2\pi \sqrt{\frac{m}{K}} \quad (42)$$

5.3 Application

The simple one dimensional pile drive model is mainly used for a fast computation of the penetration settlements at different tip and shaft resistances. Therefore, an arbitrary tip and shaft resistance can be applied to the penetrator. The simulation will determine the settlements per stroke at different penetration resistances. Thus, it will reveal the correlation between penetration rate and resistance.

Furthermore, the influence of the ratio between shaft friction and tip resistances is investigated. Therefore, the value of tip resistance and shaft friction is determined by previous DEM simulations or approximated by values of penetration resistance from cone penetration tests. In experiments by Vogelsang et al. (2017) it can be seen that the maximum resistive force at cyclic penetration is close to the resistance during a monotonic penetration as long as no friction fatigue occurs. The results of the dynamic penetration simulation in chapter 4 reveal that the maximum penetration resistance at cyclic loading depends also on the amount of penetration per stroke. It is observed that the first stroke of the hammer mass causes larger settlements and results in a larger resistance than the second stroke of the suppressor mass.

5.4 Simulation results

The numerical penetration model in Matlab provides information on the movements of the masses of the hammering mechanism and the penetrating probe. Thus, the performance of the driving mechanism can be evaluated for different resistances. The determination of the penetration resistance itself is more difficult and needs more sophisticated models, such as the DEM model for instance.

The motions of the hammer, suppressor (support) and probe during one stroke cycle can be seen in Figure 59. The resistance force was determined by DEM simulations in advance. The acceleration of the hammer mass and the reaction onto the suppressor mass at the release of the force springs can be observed at the beginning. After the hammer mass hits the tip, a displacement of the probe of 2 mm occurs. Then, the hammer mass moves upwards and oscillates with the suppressor mass. A second stroke due to the suppressor mass is produced after 70 ms. The total penetration rate of the probe due to one stroke cycle is about 3 mm.

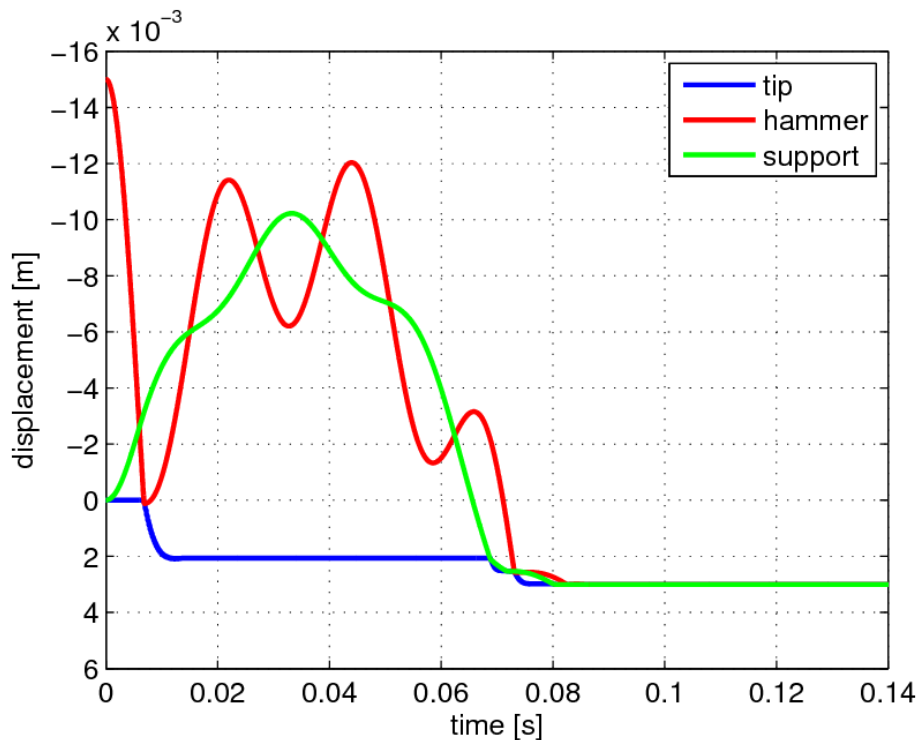


Figure 59: The relative movements of the tip, the hammer and the suppressor (support) mass from the Matlab model. (Poganski et al. 2016)

The energy that is transformed from the hammering mechanism into the movement of the probe during a stroke cycle is shown in Figure 60. The second stroke accounts for about 20 to 40 % of the total energy, so that the major contribution to the penetration is given by the first stroke of the hammer mass. The plot shows also the kinetic and potential part of the total energy. The potential energy of the springs is first transformed into a kinetic energy of the masses. Then at each stroke, the kinetic energy of a mass is transformed into a movement of the penetrator. In the end, there is a bit of energy remaining in the hammering system due to the preloading of the suppressor mass in its initial position.

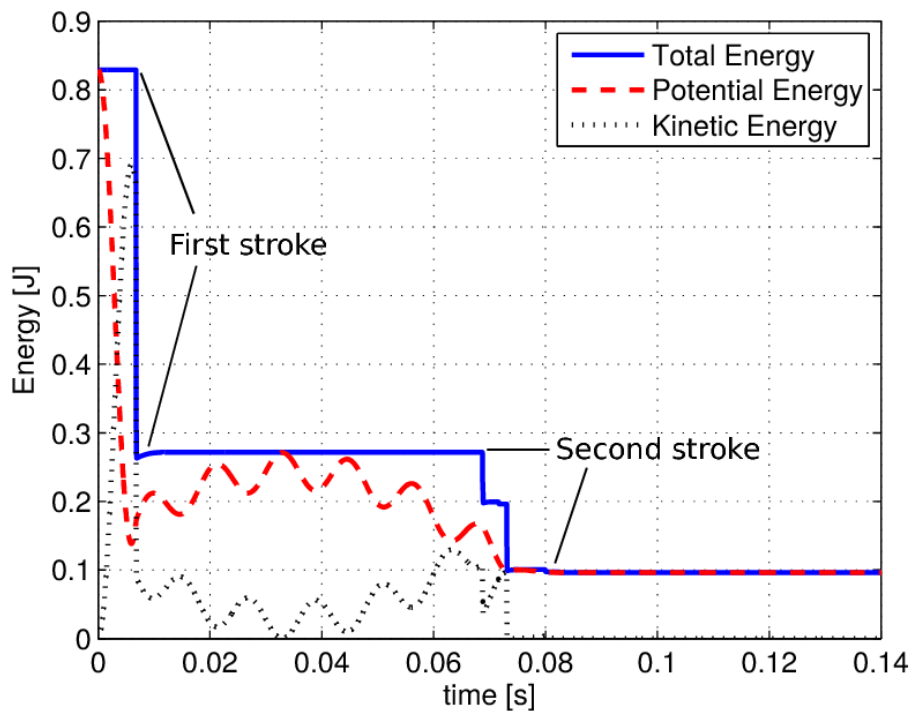


Figure 60: The kinetic energy of the support mass and the hammer mass as well as the potential energy of the break spring and the force springs over a full stroke cycle. (Poganski et al. 2016)

The influence of the ratio between shaft friction and tip friction was investigated by means of the Matlab model. Therefore, a total shaft friction of 1 to 10 % of the tip resistance were applied, where tip resistance of 80 N, 100 N and 120 N were investigated. These resistive forces were the result of cone penetration tests down to 1 m using a DEM model. From Figure 61 it can be obtained that the lower tip resistance results in a deeper penetration for the first 20 ms but causes also a larger rebound due to the lower proportional shaft friction. After the rebound happened, the difference in the total penetration due to the first stroke is only minimal. The excessive rebound due to a reduction of the relative shaft friction is clearly observed for all different tip resistances. This rebound opens a cavity in front of the tip that could collapse and reduce the penetration performance. In the Matlab model, the cavity stays open and the penetration performance is not impaired. Hence, if large rebounds occur, the Matlab model cannot represent the real soil behaviour and results in a too large penetration. The soil deformation and the tip displacement are given in Figure 62. It appears that the soil only deforms in the direction of the penetration, whereas at the rebound of the probe the soil is not affected due to unloading or collapse of the cavity. Thus, the second stroke generates a deep penetration, without any tip resistance in the beginning until the tip of the probe strikes the soil.

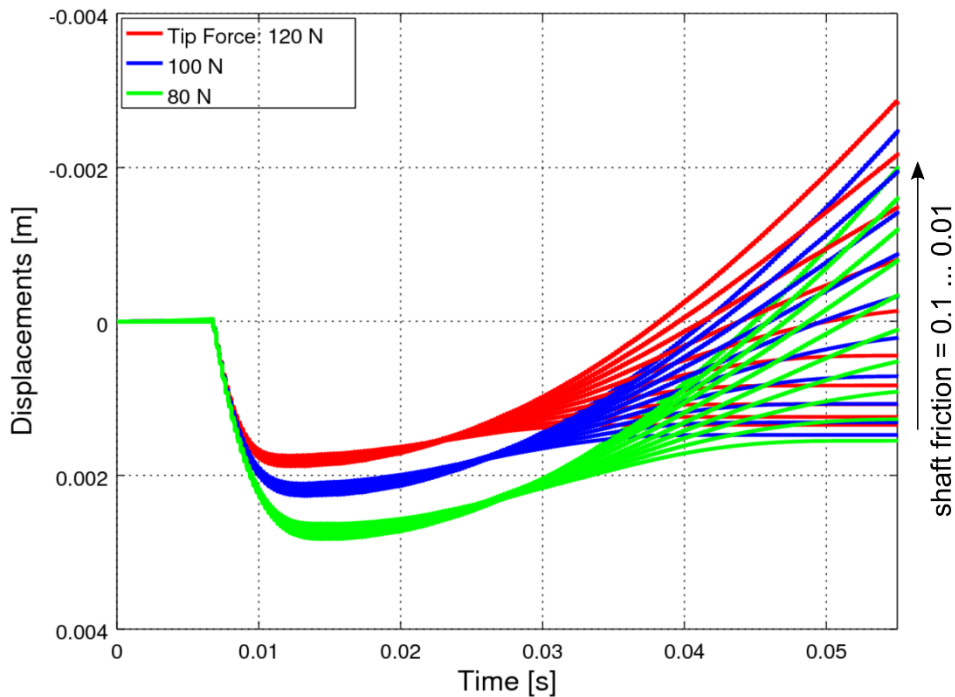


Figure 61: The penetration of HP³ due to the first stroke of the hammer mass for different tip resistance and shaft friction. The shaft friction ranges from 0.1 to 0.01 times the tip resistance.

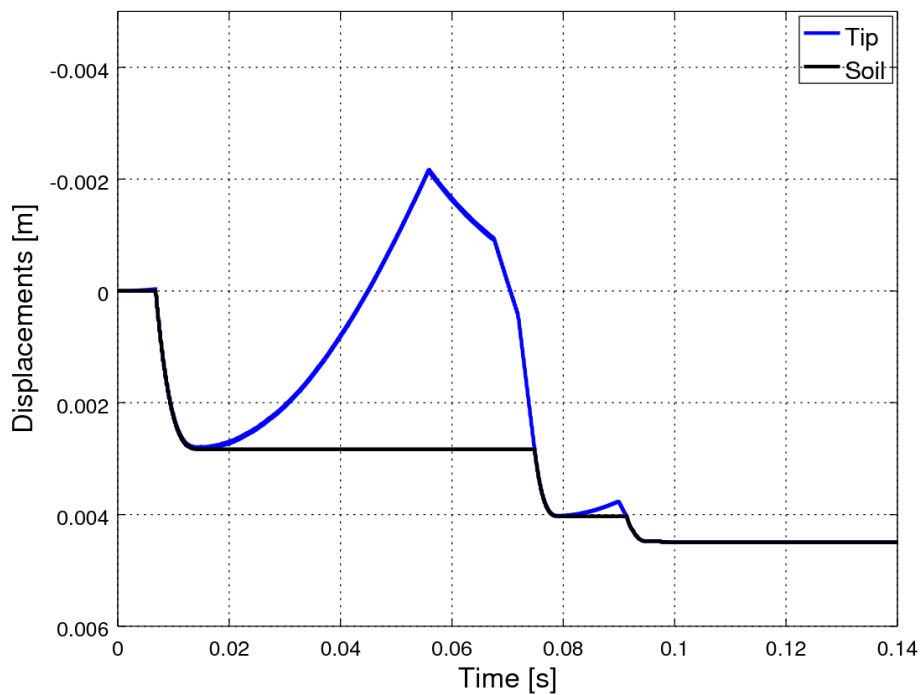


Figure 62: The displacement of the probe tip and the deformation of the soil in front of the tip for a tip resistance of 80 N and a shaft friction of 0.8 N.

The lower gravity on Mars will for sure affect the performance of the driving mechanism of HP³. Therefore, a simulation of the hammer stroke in Earth, Mars and zero gravity environment is performed for different tip resistances. A tip resistance of 120 N with a shaft friction of 20 N as well as a tip resistance of 80 N with a shaft friction of 15 N are investigated. In Figure 63 are the displacements of the penetrator for different gravitational environments. The lower gravity increases the rebound effect due to the suppressor mass moving upwards. The difference in the penetration rate is not dramatic but becomes more pronounced for lower stress levels.

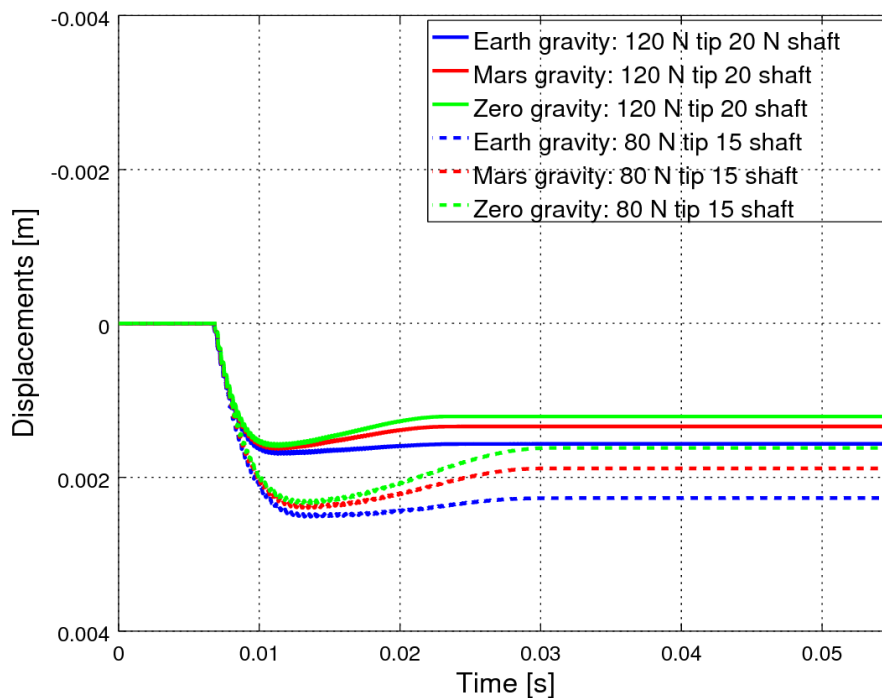


Figure 63: Penetration due to the first stroke of the hammer mass for different gravitational environments and resistances. Earth gravity: 9.81 m/s^2 , Mars gravity: 3.71 m/s^2 , Zero gravity: 0.1 m/s^2 .

5.5 Friction fatigue and the influence on the HP³ performance

In geotechnical applications, the installation procedure of piles affects the soil condition in the vicinity of the pile. Hence, also the bearing capacity is influenced by the installation method. From experimental investigations on displacement piles in sand by White et al. (2004), the impact of the installation methods on the horizontal stress was evaluated. For this purpose, the horizontal stress on a driven pile was measured by total pressure cells in several distances to the tip. It was clearly observed that a two-way cyclic loading reduced the horizontal acting stress in comparison to jacked or monotonic installed piles. The reduction of the

horizontal stress was determined to be maximum at a distance of 3 diameters from the tip. Behind that point the horizontal acting stress is almost constant. This reduction of the horizontal stress due to a dynamic penetration leads to a reduced shaft friction and is called “friction fatigue” in literature. This phenomenon appears only under cyclic loading and is not present in monotonic installed piles. White et al. (2004) investigated the appearance of friction fatigue and the dependency on the number of loading cycles for different installation types.

Further studies on friction fatigue were carried out by Basu et al. (2014). A one-dimensional finite element model was applied to understand the basic mechanisms that causes friction fatigue at cyclic loaded piles. For this purpose, the pile installation was simplified to a combination of a cylindrical cavity expansion followed by vertical shearing cycles, see Figure 64. Therefore, only the horizontal stress acting on the pile shaft was obtained. The complete load history of a dynamic penetration can be divided in 3 stages, regarding to a soil element in front of the tip. The soil element is first subjected to a cavity expansion to create a space for the penetrator. This stage is accompanied by a shearing of the soil element due to the interface friction. In the second stage, an unloading of the penetrator causes a shear unloading of the soil element and for the two-way cyclic loading even a shear load reversal. In the last stage the soil element is subjected to further shearing due to more loading cycles. As a result of the shearing reversals the soil is compacted and the horizontal stress gets reduced.

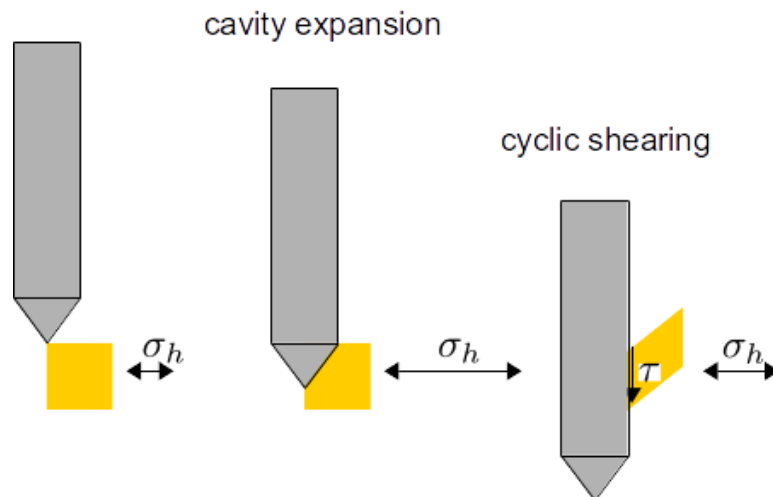


Figure 64: Simplified sketch of a dynamic pile installation

Owing to this load history, the horizontal stress near the tip is increased due to the cavity expansion and decays along the shaft with distance to the tip.

The reduction of the shaft friction may affect the performance of the HP³ penetration during its installation. It can be obtained from Figure 61 that a reduced shaft friction leads to a larger rebound after the first hammer stroke. This back motion is imposed by the upwards movement of the suppressor mass. Thus,

friction fatigue could appear and the performance of the driving mechanism may decay at a certain depth, when the ratio of tip resistance to shaft friction becomes critical.

6 Concluding remarks

In this thesis, the discrete element method was investigated with regard to its ability for the simulation of dynamic and quasistatic penetration of a probe into a dry granular material. The simulation models shall help to understand the penetration behaviour of the Heat Flow and Physical Properties Probe (HP³) from DLR. HP³ is a thermal probe that will penetrate 3 to 5 m into Martian subsurface. The instrument will be on board of the InSight lander in 2018. The numerical models enable to have a look inside the soil and allows to back-calculate physical properties of the soil from the penetration rate.

The calibration of a sandy material called Schwarzl UK4 was done to derive a first set of parameters as well as to validate the numerical method for the application. Three different tests were evaluated for the calibration procedure. An angle of repose experiment was used to derive a range of parameters for the interparticle friction. The simulation results were barely affected by the particles' stiffness and thus make it possible to determine the coefficient of interparticle friction and rolling resistance without dependency on other parameters. The angle of repose was about 32 to 33 degrees.

A triaxial shear test was performed for three different stress levels. The simulation results were analysed with respect to theoretical assumptions and compared to lab experiments. The critical state friction angle from the simulation results was determined to be about 32 degree, which is in good agreement with the slope from the angle of repose experiment. However, the comparison to the lab results showed some deviations. The lab results for a 100 kPa confining stress were in good agreement to the simulations, whereas the results for a 150 kPa confining stress had less strength in the lab test and the results for a 200 kPa confining stress showed a reduced stiffness.

Furthermore, an oedometer test was used to determine the particles' stiffness. The vertical stress of a vertically compacted sample was measured and compared. The load path contains a primary loading, an unloading and a reloading of the soil sample. The primary loading behaviour is well represented by the simulations, whereas the unloading of the sample showed too large deformations. The highly elastic behaviour at unloading is traced back to the frictional contact models in the DEM. Hence, the frictional contact models were investigated and yielded modified models that were implemented and tested. Single particle models were investigated to obtain a more realistic rolling behaviour of an irregular shaped particle. The enhanced contact models were consistently working well in 2D models, whereas in the three-dimensional case the contact models appear to have discontinuities. Hence, only a slightly modified contact model was used for the penetration simulations, where the problem of the highly elastic behaviour could not be solved.

The elastic behaviour of the contact models rise to problems as soon as an unloading of small interparticular deformations appears. This is the case for the oedometer test and the dynamic cone penetration. In the triaxial test and the quasistatic cone penetration the interparticular deformations are large enough that particles loose contact and the impact of the elastic behaviour is almost negligible.

A simulation model of the HP³ penetration was developed. The model consists of the soil domain and the hammering mechanism. In the soil domain, the resistance force on the probe is measured and applied to the hammering mechanism. In the model of the hammering mechanism, the driving velocity due to the hammering impacts is determined considering the soil resistance from the soil model. The driving velocity is then applied to the probe in the soil domain. This coupling allows for a real simulation of the penetration, where the driving mechanism depends on the soil resistance and the soil resistance depends on the penetration due to the driving mechanism. The penetration results of the fully coupled DEM model were compared to measurements and similar simulations of HP³ from literature.

Furthermore, quasistatic cone penetration tests were simulated and evaluated with the DEM. A comparison of dynamic and quasistatic driven probes was done with respect to the soil deformation, stress distribution and penetration resistance. Further CPTs with different penetration rates and in different gravitational environments were carried out. The quasistatic cone penetration was also used to identify the influence of the particle scale onto the penetration resistance.

A one-dimensional model of the dynamic simulation of HP³ was implemented in Matlab to receive fast results for specific resistance values. The model focuses on the hammering mechanism and the obtained driving force, whereas the soil response is not modelled. Therefore, the soil resistance has to be defined. The soil resistance can be estimated from cone penetration tests (CPTs) or from analytical solutions. An analytical solution for a shallow penetration under certain assumptions was derived in chapter 5.1.

The Discrete Element Method is suitable for geotechnical applications as long as the investigated case can be modelled in a small scale test. The main limitation of the method is the amount of particles that can be used. Furthermore, the particle size should not be chosen too small in order to use a feasible time step size. For the selection of a suitable DEM code, the focus should be on the implemented contact models. An implicit DEM code may also be a proper solution, since in this case the contact models for sliding and rolling resistance could be realised by step functions.

Another solution for the simulation of CPTs could be the combination of DEM and FEM. The large deformations near the penetrator could be modelled by the discrete particles whereas the surrounding soil volume is modelled by finite elements.

Therefore, discrete particles can be fixed to the element nodes in the transition area to couple both methods. Moreover, FEM based methods using particle-in-cell or remeshing techniques still remain a good possibility for the simulation of large deformations. Therefore, the work of Galavi et al. (2017) may give a solution for the simulation of a dynamic penetration using the MPM.

7 Bibliography

- Abel, D. (2010)
Regelungstechnik, Umdruck zur Vorlesung, Institut für Regelungstechnik, RWTH Aachen, 34. Auflage.
- Ai, J., Chen, J. F., Rotter, J. M. & Ooi, J. Y. (2011)
Assessment of rolling resistance models in discrete element simulations. Powder Technology, Vol. 206, No. 3, p. 269-282, doi:10.1016/j.powtec.2010.09.030.
- Basu, P., Loukidis, D., Prezzi, M. & Salgado, R. (2014)
The Mechanics of Friction Fatigue in Jacked Piles Installed in Sand, ASCE, From Soil Behavior Fundamentals to Innovations in Geotechnical Engineering: Honoring Roy E. Olson GSP 233.
- Bernhardt, M. L., O'Sullivan, C. & Biscontin, G (2015)
Effects of sample preparation methods in DEM. Geomechanics from micro to macro, Vol 1 & 2, pp. 97-102.
- Bolton, M. D., Gui, M. W., Garnier, J., Corte, J. F., Bagge, G., Laue, J. & Renzi R. (1999)
Centrifuge cone penetration tests in sand. Géotechnique, 49, No. 4, 543-552.
- Butlanska, J., Arroyo, M., Gens, A. & O'Sullivan, C. (2014)
Multi-scale analysis of cone penetration test (CPT) in a virtual calibration chamber, Can. Geotech. J., 51, 51-66.
- Ceccato, F., Beuth, L., Vermeer, P. A. & Simonini, P. (2016)
Two-phase Material Point Method applied to the study of cone penetration, Computers and Geotechnics, 80, 440-452, doi:10.1016/j.compgeo.2016.03.003.
- Christensen, U. & Knapmeyer-Endrun, B. (2016)
SEIS – Seismometer für die Mars-Mission Insight, Max-Planck-Institut für Sonnensystemforschung, [Date of access: 24.04.2017] URL: <http://www.mps.mpg.de/planetenforschung/insight-seis>
- Ciantia, M. O., Arroyo, M., Butlanska, J. & Gens, A. (2016)
DEM modelling of cone penetration tests in a double-porosity crushable granular material, Computers and Geotechnics, 73, 109-127, doi:10.1016/j.compgeo.2015.12.001.
- Deep Foundations Institute DFI & Gavin Doherty Geo Solutions GDG (2015)
COMPARISON OF IMPACT VERSUS VIBRATORY DRIVEN PILES: With focus on soil-structure interaction, report 14007-01-Rev2, [Date of access: 25.07.2017] URL: <http://www.dfi.org/update/Comparison%20of%20impact%20vs%20vibratory%20driven%20piles.pdf>

- Estrada, N., Azéma, E., Radjai, F. & Taboada, A. (2011)
Identification of rolling resistance as a shape parameter in sheared granular media. *Physical Review E : Statistical, Nonlinear and Soft Matter Physics*, American Physical Society, 2011, Vol. 84, pp.011306.
- Falagush, O., McDowell, G. R. & Yu, H. S. (2015)
Discrete element modelling of cone penetration tests incorporating particle shape and crushing, *Int. J. Geomech.*, doi:10.1061/(ASCE)GM.1943-5622.0000463, 04015003.
- Galavi, V., Beuth, L., Coelho, B. Z., Tehrani, F. S., Hölscher, P. & Tol, F. V. (2017)
Numerical simulation of pile installation in saturated sand using material point method, *Procedia Engineering*, vol. 175, pp. 72-79, doi:10.1016/j.proeng.2017.01.027.
- Golombek, M., Kipp, D., Warner, N. et al. (2016)
Selection of the InSight Landing Site, *Space Sci Rev*, doi:10.1007/s11214-016-0321-9.
- Grabe, J. & Pucker, T (2013)
Numerisch gestützte Entwicklung von Geräten und Verfahren des Spezialtiefbaus, Beiträge zum 12. Geotechnik-Tag in München, *Geotechnik und industrielle Verfahren*, pp. 33-46.
- Hamad, F., Giridharan, S. & Moormann, C. (2017)
A penalty function method for modelling frictional contact in MPM. 1 st International Conference on the Material Point Method, MPM 2017, *Procedia Engineering*, vol. 175, pp. 116-123, doi:10.1016/j.proeng.2017.01.038.
- Holmen, J. K., Olovsson, L. & Borvik, T. (2017)
Discrete modelling of low-velocity penetration in sand, *Computers and Geotechnics*, 86, 21-32, doi:10.1016/j.compgeo.2016.12.021.
- Issam, K. J. (2013)
Formulation of a Dynamic Material Point Method (MPM) for Geomechanical Problems, Doctoral dissertation, Institut für Geotechnik, Universität Stuttgart.
- Jiang, M., Shen, Z. & Wang, J. (2015)
A novel three-dimensional contact model for granulates incorporating rolling and twisting resistance, *Computers and Geotechnics*, 65, 147-163, doi:10.1016/j.compgeo.2014.12.011.
- Kloss, C., Goniva, C., Hager, A., Amberger, S. & Pirker, S. (2012)
Models, algorithms and validation for opensource DEM and CFD-DEM, *Computational Fluid Dynamics, An Int. J.* 2012, Vol. 12, No.2/3, pp140 - 152.

- Kulak, R. F. & Bojanowski, C. (2011)
Modeling of Cone Penetration Test Using SPH and MM-ALE Approaches, 8th European LS-DYNA Users Conference, Strasbourg.
- Lichtenheldt, R. & Krömer, O. (2016)
Soil modelling for InSight's HP³-Mole: From highly accurate particle-based towards fast empirical models. In: Earth & Space, ASCE.
- Lichtenheldt, R., Schäfer, B. & Krömer, O. (2014)
Hammering beneath the surface of Mars – modelling and simulation of the impact-driven locomotion of the hp3-mole by coupling enhanced multi-body dynamics and discrete element method. In Shaping the future by engineering: 58th Ilmenau Scientific Colloquium IWK, URN (Paper): urn:nbn:de:gbv:ilm1-2014iwk-155:2, Technische Universität Ilmenau, 08 – 12 September 2014.
- Lichtenheldt, R. & Schäfer, B. (2013)
Planetary rover locomotion on soft granular soils – efficient adaption of the rolling behaviour of nonspherical grains for discrete element simulations. In 3rd International Conference on Particle-Based Methods, S.807-818, ISBN 978-84-941531-8-1, Stuttgart.
- LIGGGHTS®-PUBLIC documentation (2017)
Version 3.X, [Date of access: 16.10.2017] URL: <https://www.cfdem.com/media/DEM/docu/Manual.html>
- Moore, H. J., Bickler, D. B., Crisp, J. A., Eisen, H. J., Gensler, J. A., Haldemann, A. F. C., Matijevic, J. R., Reid, L. K. & Pavlics, F. (1999)
Soil-like deposits observed by Sojourner, the Pathfinder rover, J. Geophys. Res., 104(E4), 8729-8746, doi:10.1029/1998JE900005.
- Moore, H. J. & Jakosky, B. M. (1989)
Viking Landing Sites, Remote-Sensing Observations, and Physical Properties of Martian Surface Materials, ICARUS, 81, 164-184.
- NASA/JPL-Caltech (2015a)
Artist's Concept of InSight Lander on Mars, [Date of access: 23.07.2017] URL: <https://insight.jpl.nasa.gov/images.cfm?ImageID=8501>
- NASA/JPL-Caltech (2015b)
Landing Area Narrowed for 2016 InSight Mission to Mars, [Date of access: 23.07.2017] URL: <https://insight.jpl.nasa.gov/images.cfm?ImageID=8310>
- O'Sullivan, C. (2011)
Particle-Based Discrete Element Modeling: Geomechanics Perspective, Int. J. Geomech., 449-464, doi:10.1061/(ASCE)GM.1943-5622.0000024.
- Perko, H., Nelson, J. & Green, J. (2006)
Mars Soil Mechanical Properties and Suitability of Mars Soil Simulants, J. Aerosp. Eng., 10.1061/(ASCE)0893-1321(2006)19:3(169), 169-176.

- Pike, W. T., Stauffer, U., Hecht, M. H., Goetz, W., Parrat, D., Sykulska-Lawrence, H., Vijendran, S. & Madsen, M. B. (2011)
Quantification of the dry history of the Martian soil inferred from in situ microscopy, *Geophys. Res. Lett.*, 38, L24201, doi:10.1029/2011GL049896.
- Poganski, J., Kömle, N. I., Kargl, G., Schweiger, H. F., Grott, M., Spohn, T., Krömer, O., Krause, C., et al. (2016)
Extended Pile Driving Model to Predict the Penetration of the Insight/HP3 Mole into the Martian Soil. *Space Science Reviews*, doi: 10.1007/s11214-016-0302-z.
- Shaw, A., Arvidson, R. E., Bonitz, R., Carsten, J., Keller, H. U., Lemmon, M. T., Mellon, M. T., Robinson, M. & Trebi-Ollennu, A. (2009)
Phoenix soil physical properties investigation, *J. Geophys. Res.*, 114, E00E05, doi:10.1029/2009JE003455.
- Smith, E. A. L. (1962)
Pile driving analysis by the wave equation, *Am. Soc. Civil Eng. (ASCE) Trans.* 127, 1145-1193.
- Spohn, T. (2013)
Heat Flow and Physical Properties Package HP³, DLR, Institute of Planetary Research.
- Sullivan, R., Anderson, R., Biesiadecki, J., Bond, T. & Stewart, H. (2011)
Cohesion, friction angles, and other physical properties of Martian regolith from Mars Exploration Rover wheel trenches and wheel scuffs, *J. Geophys. Res.*, 116, E02006, doi:10.1029/2010JE003625.
- Sulsky, D., Chen, Z. & Schreyer, H. L. (1993)
A particle method for history-dependent materials, Technical Report SAND93-7044, Sandia National Laboratories, Albuquerque U.S.
- Sulsky, D. & Schreyer, H. L. (1996)
Axisymmetric form of the material point method with applications to upsetting and Taylor impact problems, *Computer Methods in Applied Mechanics and Engineering*, 139(1-4):409-429.
- Susila, E. & Hryciw, R. D. (2003)
Large displacement FEM modelling of the cone penetration test (CPT) in normally consolidated sand, *Int. J. Numer. Anal. Meth. Geomech.*, 27, 585-602, doi:10.1002/nag.287.
- Terzaghi, K. (1943)
Theoretical Soil Mechanics. John Wiley & Sons, Inc., New York, doi:10.1002/9780470172766
- Verlet, L. (1967)
Computer “Experiments” on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules, *Phys. Rev.*, Vol. 159, Iss. 1, 98-103.

- Vogelsang, J., Huber, G. & Triantafyllidis, T. (2017)
Stress Paths on Displacement Piles During Monotonic and Cyclic Penetration. *Holistic Simulation of Geotechnical Installation Processes*, 82, Springer International Publishing, pp. 29-52, doi: 10.1007/978-3-319-52590-7_2.
- White, D. J. & Lehane, B. M. (2004)
Friction fatigue on displacement piles in sand, *Géotechnique* 54, No. 10, 645-658.
- Wieckowski, Z., Youn, S.-K. & Yeon, J.-H. (1999)
A particle-in-cell solution to the silo discharging problem, *International Journal for Numerical Methods in Engineering*, 45(9):1203-1225.
- Williams, J.-P., Nimmo, F., Moore, W. B. & Paige, D. A. (2008)
The formation of Tharsis on Mars: What the line-of-sight gravity is telling us, *J. Geophys. Res.*, 113, E10011, doi:10.1029/2007JE003050.
- Bardenhagen, S. G., Brackbill, J. U. & Sulsky, D. (2000)
The material-point method for granular materials. *Computer Methods in Applied Mechanics and Engineering*, 187, (3-4):529-541.
- Zöhrer, A. (2006)
Laboratory Experiments and Numerical Modelling of Cone Penetration Tests into various Martian Soil Analogue Materials, Doctoral dissertation, Institut für Bodenmechanik und Grundbau, Technische Universität Graz.

8 Appendix

8.1 Simulation model structure

The structure of the different calibration and penetration models is kept very similar. The complete simulation, combining the filling process, consolidation and the execution of the test, is loaded in the *in.run* file. For some models, e.g. the oedometer test, the *in.run* file defines already variables for the simulation to avoid changes in the substructure files and to allow a quick adjustment of the main parameters. In general, the important parameters that specify the soil behaviour are implemented in the *in.variables* file, where parameters that are defined already in the *in.run* file are commented out. Furthermore, the parameters that define the geometry of the boundaries, the particle volume fraction for insertion and the scale size of the particles are set within the *in.variables* file. There are also a few parameters defined in the *in.variables* file that are just important for special models and sets the load steps for the oedometer simulation or the parameters for the control unit of the stress controlled walls. The stress controlled walls are implemented by the *fix mesh/surface/stress/servo* command and requires a set parameters for the PID controller. Those parameters are the proportional, integral and differential constant for the PID controller.

All simulations begin with a file that generates the particle filling. Therefore, a particle radius expansion method is used to generate a homogeneous structure, see chapter 2.8. The file that models the filling process is named as '*in.fill*' or for some cases with extension '*in.fill_...*'. The *in.fill* file generates a particle bedding, which is different for each model due to differences in the geometry of the boundaries. A restart file is generated at the end of the *in.fill* file to allow for a restart of the simulation from the current time step. The restart file is then loaded for the main simulation of the test. In some cases a separate consolidation file is used between the filling and the execution of the test, where usually an appropriate consolidation is already considered in the *in.fill* file.

The generation of the particle size distribution is done within the *in.verteilunlinear* file. The particle size distribution is created by 3 radii sizes with corresponding mass fractions. In between those 3 radii sizes there are further radii and mass fractions defined on a linear interpolation to receive a smooth distribution. For simulations with more than one particle size distribution, the *in.verteilunlineardevide* file is used. This is necessary for simulations with different scales of the particle size distribution.

The *in.getdensity* file can be used to determine the current bulk density of the soil skeleton by means of the Voronoi tessellation. The file is suited for the geometry of the 3D cone penetration model. The *in.getdensity* requires the restart file of the

filling and computes the density after the filling process. The value of the density is stored in a text file called *info.txt*.

The important output files for the different simulation models are

- Dynamic Cone Penetration: outputcheck.txt, molepos.txt
- Quasistatic Cone Penetration: outputcheck.txt
- Angle of Repose Test: results.txt
- Oedometer Test: results/results_oed...
- Triaxial Shear Test: results.txt

The content of the output files is different for the most models, therefore a list of the content can be found in Table 4.

Table 4: Output files and the corresponding content in detail

	Dynamic Cone Penetration: molepos.txt	Dynamic/Quasistatic Cone Penetration: outputcheck.txt	Angle of Repose Test: results.txt	Oedometer Test: results/results_oed...	Triaxial Shear Test: results.txt
1	Time Step	Time Step	Time Step	Time Step	Time Step
2	Tip Displacement in x [m]	Penetration Resistance in x [N]	Interparticle friction	Stress on Stamp [Pa]	Vertical Force at Bottom [N]
3	Tip Displacement in y [m]	Penetration Resistance in y [N]	Interparticle rolling resistance	Relative Strain of Stamp [%]	Cross Sectional Area [m ²]
4	Tip Displacement in z [m]	Penetration Resistance in z [N]	Mean Slope Value		Total Position of Bottom Wall [m]
5	Penetration Resistance in x [N]	Penetration Velocity in x [m/s]	Slope Value at Top		Horizontal Confining stress [Pa]
6	Penetration Resistance in y [N]	Penetration Velocity in y [m/s]	Slope Value at Centre		Total Volume [m ³]

7	Penetration Resistance in z [N]	Penetration Velocity in z [m/s]	Slope Value at Bottom		Relative Axial Strain [%]
8	Incremental Time Step	Incremental Time Step			Area of Wall A [m ³]
9	Incremental Hammer Displacement [m]	Total Position of Penetrator in x [m]			Area of Wall C [m ³]
10	Incremental Suppressor Displacement [m]	Total Position of Penetrator in y [m]			
11	Incremental Penetrator Displacement [m]	Total Position of Penetrator in z [m]			

8.2 Modified rolling model sbjp

#The SBJP model was mainly used for the simulation of HP3

```
#ifdef ROLLING_MODEL
```

```
ROLLING_MODEL(ROLLING_SBJP,sbjp,5)
```

```
#else
```

```
#ifndef ROLLING_MODEL_SBJP_H_
```

```
#define ROLLING_MODEL_SBJP_H_
```

```
#include "contact_models.h"
```

```
#include <algorithm>
```

```
#include "math.h"
```

```
#include "domain.h"
```

```
#include "math_extra_liggghts.h"
```

```
namespace LIGGGHTS {
```

```
namespace ContactModels
{
    using namespace LAMMPS_NS;

    template<>
    class RollingModel<ROLLING_SBJP> : protected Pointers
    {
    public:

        static const int MASK = CM_CONNECT_TO_PROPERTIES |
        CM_SURFACES_INTERSECT | CM_SURFACES_CLOSE;

        RollingModel(class LAMMPS * Imp, IContactHistorySetup * hsetup, class
        ContactModelBase *) :

            Pointers(Imp), coeffRollFrict(NULL)
        {
            history_offset = hsetup->add_history_value("rollangle", "1");
            hsetup->add_history_value("rollangle_y", "1");
            hsetup->add_history_value("rollangle_z", "1");
            hsetup->add_history_value("roll_flag", "0");
            hsetup->add_history_value("T_rollangle_x", "1");
            hsetup->add_history_value("T_rollangle_y", "1");
            hsetup->add_history_value("T_rollangle_z", "1");
            hsetup->add_history_value("roll_yield_flag", "0");
            hsetup->add_history_value("twist_yield_flag", "0");
        }

        void registerSettings(Settings&) {}

        void connectToProperties(PropertyRegistry & registry) {

            registry.registerProperty("coeffRollFrict",
            &MODEL_PARAMS::createCoeffRollFrict);
```



```

registry.connect("coeffRollFrict", coeffRollFrict,"rolling_model sbjp");

registry.registerProperty("coeffFrict",
&MODEL_PARAMS::createCoeffFrict);

registry.connect("coeffFrict", coeffFrict,"rolling_model sbjp");

// error checks on coarsegraining

if(force->cg_active())

    error->cg(FLERR,"rolling model sbjp");

}

void surfacesIntersect(SurfacesIntersectData & sidata, ForceData & i_forces,
ForceData & j_forces)

{

    double
r_torque[3],T_torque[3],r_coef,wr_n_i[3],wr_n_j[3],wr_t_i[3],wr_t_j[3];

    vectorZeroize3D(r_torque);

    vectorZeroize3D(T_torque);

    if(sidata.contact_flags)                *sidata.contact_flags                |=
CONTACT_ROLLING_MODEL;

    const double radi = sidata.radi;

    const double radj = sidata.radj;

    double reff=sidata.is_wall ? sidata.radi : (radi*radj/(radi+radj));

#ifdef SUPERQUADRIC_ACTIVE_FLAG

    if(sidata.is_non_spherical)

        reff = MathExtraLiggghtsSuperquadric::get_effective_radius(sidata);

#endif

    if(sidata.is_wall) {

        const double wr1 = sidata.wr1;

        const double wr2 = sidata.wr2;

```

```

const double wr3 = sidata.wr3;

const double radius = sidata.radi;

double r_inertia;

if (domain->dimension == 2) r_inertia = 1.5*sidata.mi*radius*radius;

else r_inertia = 1.4*sidata.mi*radius*radius;

calcRollTorque(r_torque,T_torque,sidata,reff,wr1,wr2,wr3,r_inertia,r_coef);

/*

const double wr_dot_delta_i = sidata.en[0]*wr1 + sidata.en[1]*wr2 +
sidata.en[2]*wr3; //projection

vectorScalarMult3D(sidata.en, wr_dot_delta_i, wr_n_i);

wr_t_i[0]=wr1 -wr_n_i[0];

wr_t_i[1]=wr2 -wr_n_i[1];

wr_t_i[2]=wr3 -wr_n_i[2];

vectorCopy3D(wr_n_i, wr_n_j);

vectorCopy3D(wr_t_i, wr_t_j);*/

} else {

double wr_roll[3];

const int i = sidata.i;

const int j = sidata.j;

const double * const * const omega = atom->omega;

const double r_inertia_red_i = sidata.mi*radi*radi;

const double r_inertia_red_j = sidata.mj*radj*radj;

double r_inertia;

if (domain->dimension == 2) r_inertia = 1.5 * r_inertia_red_i *
r_inertia_red_j/(r_inertia_red_i + r_inertia_red_j);

else r_inertia = 1.4 * r_inertia_red_i * r_inertia_red_j/(r_inertia_red_i +
r_inertia_red_j);

```

```

// relative rotational velocity

vectorSubtract3D(omega[i],omega[j],wr_roll);

calcRollTorque(r_torque,T_torque,sidata,reff,wr_roll[0],wr_roll[1],wr_roll[2],r_i
nertia,r_coef);

/*

    const double wr_dot_delta_i = vectorDot3D(omega[i],sidata.en);
//projection

    vectorScalarMult3D(sidata.en, wr_dot_delta_i, wr_n_i);

    vectorSubtract3D(omega[i],wr_n_i, wr_t_i);

    const double wr_dot_delta_j = vectorDot3D(omega[j],sidata.en);
//projection

    vectorScalarMult3D(sidata.en, wr_dot_delta_j, wr_n_j);

    vectorSubtract3D(omega[j],wr_n_j, wr_t_j);*/

/*

    const double T_transmit=0.6;

    if(vectorMag3D(omega[i])<vectorMag3D(omega[j])){

        i_forces.delta_torque[0] ==
T_transmit*(r_torque[0]+T_torque[0]);//+r_coef*wr_t_i[0]);

        i_forces.delta_torque[1] ==
T_transmit*(r_torque[1]+T_torque[1]);//+r_coef*wr_t_i[1]);

        i_forces.delta_torque[2] ==
T_transmit*(r_torque[2]+T_torque[2]);//+r_coef*wr_t_i[2]);

        j_forces.delta_torque[0] += r_torque[0]+T_torque[0];//+r_coef*wr_t_j[0];

        j_forces.delta_torque[1] += r_torque[1]+T_torque[1];//+r_coef*wr_t_j[1];

        j_forces.delta_torque[2] += r_torque[2]+T_torque[2];//+r_coef*wr_t_j[2];

    }else{

        i_forces.delta_torque[0] ==
(r_torque[0]+T_torque[0]);//+r_coef*wr_t_i[0]);

```

```

        i_forces.delta_torque[1]                                -=
(r_torque[1]+T_torque[1]);//+r_coef*wr_t_i[1]);

        i_forces.delta_torque[2] -= (r_torque[2]+T_torque[2]);//+r_coef*wr_t_i[2]);

        j_forces.delta_torque[0]                                +=
T_transmit*(r_torque[0]+T_torque[0]);//+r_coef*wr_t_j[0];

        j_forces.delta_torque[1]                                +=
T_transmit*(r_torque[1]+T_torque[1]);//+r_coef*wr_t_j[1];

        j_forces.delta_torque[2]                                +=
T_transmit*(r_torque[2]+T_torque[2]);//+r_coef*wr_t_j[2];

        }*/

    }

    i_forces.delta_torque[0] -= (r_torque[0]+T_torque[0]);//+r_coef*wr_t_i[0]);
    i_forces.delta_torque[1] -= (r_torque[1]+T_torque[1]);//+r_coef*wr_t_i[1]);
    i_forces.delta_torque[2] -= (r_torque[2]+T_torque[2]);//+r_coef*wr_t_i[2]);
    j_forces.delta_torque[0] += r_torque[0]+T_torque[0];//+r_coef*wr_t_j[0];
    j_forces.delta_torque[1] += r_torque[1]+T_torque[1];//+r_coef*wr_t_j[1];
    j_forces.delta_torque[2] += r_torque[2]+T_torque[2];//+r_coef*wr_t_j[2];

}

void surfacesClose(SurfacesCloseData & sdata, ForceData&, ForceData&)

{

    if(sdata.contact_flags)                *sdata.contact_flags                &=
~CONTACT_ROLLING_MODEL;

    double * const c_history = &sdata.contact_history[history_offset];

    c_history[0] = 0.0; // this is the r_torque_old
    c_history[1] = 0.0; // this is the r_torque_old
    c_history[2] = 0.0; // this is the r_torque_old
    c_history[3] = rand() % 101; //roll_flag [0, 1]

```

```

    c_history[4] = 0.0; // this is the T_torque_old
    c_history[5] = 0.0; // this is the T_torque_old
    c_history[6] = 0.0; // this is the T_torque_old
    c_history[7] = 0; //total angle shear
    c_history[8] = 0;//0.0; //k rotational
}

void beginPass(SurfacesIntersectData&, ForceData&, ForceData&){}

void endPass(SurfacesIntersectData&, ForceData&, ForceData&){}

private:

    double ** coeffRollFrict;

    double ** coeffFrict;

    int history_offset;

    inline void calcRollTorque(double (&r_torque)[3],double (&T_torque)[3],const
SurfacesIntersectData & sidata,double reff,double wr1,double wr2,double wr3,
double r_inertia, double (&r_coef)) {

        double wr_n[3],wr_t[3];

        const double enx = sidata.en[0];

        const double eny = sidata.en[1];

        const double enz = sidata.en[2];

        const double dt = update->dt;

        double * const c_history = &sidata.contact_history[history_offset]; // requires
Style::TANGENTIAL == TANGENTIAL_HISTORY

        const double rmu = coeffRollFrict[sidata.itype][sidata.jtype];

        const double xmu = coeffFrict[sidata.itype][sidata.jtype];

        // remove normal (torsion) part of relative rotation

        // use only tangential parts for rolling torque

        const double wr_dot_delta = wr1*enx+ wr2*eny + wr3*enz;

```



```
c_history[2] *= qxfactor;

c_history[7] = 0;

}else if(c_history[7]==3 && quad_oldrollmag>quad_rollmag){

c_history[0] = 0;

c_history[1] = 0;

c_history[2] = 0;

c_history[7] = 0;

}*/

r_torque[0] = kr*c_history[0];

r_torque[1] = kr*c_history[1];

r_torque[2] = kr*c_history[2];

// limit max. torque

const double r_torque_mag = vectorMag3D(r_torque);

const double r_torque_max = fabs(sidata.Fn)*reff*rmu;//

if(r_torque_mag > r_torque_max)

{

if(r_torque_mag != 0.0){

const double factor = r_torque_max / r_torque_mag;

r_torque[0] *= factor;

r_torque[1] *= factor;

r_torque[2] *= factor;

// save rolling torque due to spring

c_history[0] = r_torque[0]/kr;

c_history[1] = r_torque[1]/kr;
```

```
c_history[2] = r_torque[2]/kr;

/*
if(c_history[7] == 2){
    c_history[7] = 3;
} else if (c_history[7] == 0){
    c_history[7] = 2;
}*/

} else {

// dashpot
r_coef = 2.0*sqrt(r_inertia*kr);
r_torque[0] += r_coef*wr_t[0];
r_torque[1] += r_coef*wr_t[1];
r_torque[2] += r_coef*wr_t[2];

const double c_r_torque_mag = vectorMag3D(r_torque);
if(c_r_torque_mag > r_torque_max && c_r_torque_mag!=0)
{
    const double factorial = r_torque_max/c_r_torque_mag;
    vectorScalarMult3D(r_torque, factorial);
    c_history[7]=1;
}
}

//=====TORSIONAL PART

double r_coef_twist;

//double item = 1.0-sidata.deltan/(2*reff);
```



```

    const          double          T_torque_max          =
fabs(sidata.Fn)*xmu*0.65*reff*rmu;//sqrt(1.0-iterm*iterm);

    const double krT = sidata.kt*reff*reff;

    double dr_torqueT[3];

        double dt_angle[3];

    vectorScalarMult3D(wr_n,dt,dt_angle);

    const          double          quad_oldtwistmag          =
sqrt(c_history[4]*c_history[4]+c_history[5]*c_history[5]+c_history[6]*c_history
[6]);

    c_history[4] += dt_angle[0];

    c_history[5] += dt_angle[1];

    c_history[6] += dt_angle[2];

        /*

    const          double          quad_twistmag          =
sqrt(c_history[4]*c_history[4]+c_history[5]*c_history[5]+c_history[6]*c_history
[6]);

    if(c_history[8]==1 && quad_oldtwistmag<quad_twistmag){

        const double qqfactor = 1;//quad_oldtwistmag/quad_twistmag;

        c_history[4] *= qqfactor;

        c_history[5] *= qqfactor;

        c_history[6] *= qqfactor;

        c_history[8]=0;

    }else if(c_history[8]==3 && quad_oldtwistmag>quad_twistmag){

        c_history[4] = 0;

        c_history[5] = 0;

        c_history[6] = 0;

        c_history[8]=0;

```

```
    }*/

    T_torque[0] = krT*c_history[4];
T_torque[1] = krT*c_history[5];
T_torque[2] = krT*c_history[6];
const double T_torque_mag = vectorMag3D(T_torque);
if(T_torque_mag > T_torque_max)
{
    if(T_torque_mag != 0.0){
const double Tfactor = T_torque_max / T_torque_mag;
T_torque[0] *= Tfactor;
T_torque[1] *= Tfactor;
T_torque[2] *= Tfactor;
c_history[4] = T_torque[0]/krT;
c_history[5] = T_torque[1]/krT;
c_history[6] = T_torque[2]/krT;
/*
if(c_history[8] == 2){
    c_history[8] = 3;
    }else if (c_history[8] == 0){
    c_history[8] = 2;
    }
*/
    }else{T_torque[0]=T_torque[1]=T_torque[2]=0.0;}
r_coef_twist=0.0;
    }else{
```

```

        r_coef_twist=0.0;

        if(T_torque_max!=0){

r_coef_twist = 0.3* 2.0 * sqrt(r_inertia*krT);

        }

    }

        T_torque[0] += r_coef_twist*wr_n[0];

T_torque[1] += r_coef_twist*wr_n[1];

T_torque[2] += r_coef_twist*wr_n[2];

//=====end of torsional part

double c_T_torque_mag = vectorMag3D(T_torque);

if(c_T_torque_mag > T_torque_max && c_T_torque_mag!=0)

{

        double factorialT = T_torque_max/c_T_torque_mag;

        vectorScalarMult3D(T_torque, factorialT);

        c_history[8]=1;

    }

}

};

}

}

#endif // ROLLING_MODEL_SBJP_H_

#endif

```

8.3 Modified rolling model stone2

```

#ifdef ROLLING_MODEL

ROLLING_MODEL(ROLLING_STONE2,stone2,6)

```

```
#else

#ifndef ROLLING_MODEL_STONE2_H_
#define ROLLING_MODEL_STONE2_H_

#include "contact_models.h"

#include <algorithm>

#include "math.h"

#include "domain.h"

#include "math_extra_liggghts.h"

namespace LIGGGHTS {
namespace ContactModels
{
using namespace LAMMPS_NS;

template<>
class RollingModel<ROLLING_STONE2> : protected Pointers
{
public:
    static const int MASK = CM_CONNECT_TO_PROPERTIES |
CM_SURFACES_INTERSECT | CM_SURFACES_CLOSE;

    RollingModel(class LAMMPS * lmp, IContactHistorySetup * hsetup, class
ContactModelBase *) : Pointers(lmp), coeffRollFrict(NULL), coeffFrict(NULL),
treach_flag(false)//, Ref_T(0.0), kr_O(0.0)
    {
        history_offset = hsetup->add_history_value("r_torquex_old", "1");
        hsetup->add_history_value("r_torquey_old", "1");
        hsetup->add_history_value("r_torquez_old", "1");
        hsetup->add_history_value("the_flag", "1");
        hsetup->add_history_value("kr", "1");
    }
};
};
};
```

```
hsetup->add_history_value("kr_O", "1");
hsetup->add_history_value("firsttouch", "1");
hsetup->add_history_value("kt", "1");
hsetup->add_history_value("r_torquey_direction", "1");
hsetup->add_history_value("r_torquez_direction", "1");
hsetup->add_history_value("transmit_torque_i", "1");
hsetup->add_history_value("transmit_torque_j", "1");
hsetup->add_history_value("T_torquex_old", "1");
hsetup->add_history_value("T_torquey_old", "1");
hsetup->add_history_value("T_torquez_old", "1");
}

void registerSettings(Settings&) {}

void connectToProperties(PropertyRegistry & registry) {

    registry.registerProperty("coeffRollFrict",
&MODEL_PARAMS::createCoeffRollFrict);

    registry.connect("coeffRollFrict", coeffRollFrict, "rolling_model stone2");

    registry.registerProperty("coeffFrict",
&MODEL_PARAMS::createCoeffFrict);

    registry.connect("coeffFrict", coeffFrict, "tangential_model history");

    // error checks on coarsegraining
    if(force->cg_active())

        error->cg(FLERR, "rolling model stone2");

}

void surfacesIntersect(SurfacesIntersectData & sidata, ForceData & i_forces,
ForceData & j_forces)

{
```

```

    double r_torque[3],T_torque[3], transmittingratio_i, transmittingratio_j,
    r_coef,r_coef_twist,wr_n_i[3],wr_n_j[3],wr_t_i[3],wr_t_j[3];
    //tipreach_flag=false;

    vectorZeroize3D(r_torque);

    vectorZeroize3D(T_torque);

    if(sidata.contact_flags) *sidata.contact_flags |=
CONTACT_ROLLING_MODEL;

    #ifdef SUPERQUADRIC_ACTIVE_FLAG

    if(sidata.is_non_spherical)

        reff = MathExtraLiggghtsSuperquadric::get_effective_radius(sidata);

#endif

    if(sidata.is_wall) {

        const double wr1 = sidata.wr1;

        const double wr2 = sidata.wr2;

        const double wr3 = sidata.wr3;

        const double radius = sidata.radi;

        double r_inertia;

        if (domain->dimension == 2) r_inertia = 1.5*sidata.mi*radius*radius;

        else r_inertia = 1.4*sidata.mi*radius*radius;

    }

    calcRollTorque(r_torque,T_torque,sidata,radius,wr1,wr2,wr3,r_inertia,transmitti
ngratio_i,transmittingratio_j,r_coef,r_coef_twist);

    transmittingratio_i=1;

    const double wr_dot_delta_i = sidata.en[0]*wr1 + sidata.en[1]*wr2 +
sidata.en[2]*wr3; //projection

    vectorScalarMult3D(sidata.en, wr_dot_delta_i, wr_n_i);

    wr_t_i[0]=wr1 -wr_n_i[0];

    wr_t_i[1]=wr2 -wr_n_i[1];

```

```

        wr_t_i[2]=wr3 -wr_n_i[2];

        vectorCopy3D(wr_n_i, wr_n_j);

        vectorCopy3D(wr_t_i, wr_t_j);

    } else {

        double wr_roll[3];

        const int i = sidata.i;

        const int j = sidata.j;

        const double radi = sidata.radi;

        const double radj = sidata.radj;

        const double reff = sidata.is_wall ? radi :
min(radi,radj);//(radi*radj/(radi+radj));

        const double * const * const omega = atom->omega;

        const double r_inertia_red_i = sidata.mi*radi*radi;

        const double r_inertia_red_j = sidata.mj*radj*radj;

        double r_inertia;

        if (domain->dimension == 2) r_inertia = 1.5 * r_inertia_red_i *
r_inertia_red_j/(r_inertia_red_i + r_inertia_red_j);

        else r_inertia = 1.4 * r_inertia_red_i * r_inertia_red_j/(r_inertia_red_i +
r_inertia_red_j);

        // relative rotational velocity

        vectorSubtract3D(omega[i],omega[j],wr_roll);

calcRollTorque(r_torque,T_torque,sidata,reff,wr_roll[0],wr_roll[1],wr_roll[2],r_i
nertia,transmittingratio_i,transmittingratio_j,r_coef,r_coef_twist);

        transmittingratio_i=1;

        const double wr_dot_delta_i = vectorDot3D(omega[i],sidata.en); //projection

        vectorScalarMult3D(sidata.en, wr_dot_delta_i, wr_n_i);

```

```

vectorSubtract3D(omega[i],wr_n_i, wr_t_i);

const double wr_dot_delta_j = vectorDot3D(omega[j],sidata.en); //projection
vectorScalarMult3D(sidata.en, wr_dot_delta_j, wr_n_j);

vectorSubtract3D(omega[j],wr_n_j, wr_t_j);

}

i_forces.delta_torque[0] -= (r_torque[0]*transmittingratio_i + T_torque[0] +
r_coef*wr_t_i[0] + r_coef_twist*wr_n_i[0]);

i_forces.delta_torque[1] -= (r_torque[1]*transmittingratio_i + T_torque[1] +
r_coef*wr_t_i[1] + r_coef_twist*wr_n_i[1]);

i_forces.delta_torque[2] -= (r_torque[2]*transmittingratio_i + T_torque[2] +
r_coef*wr_t_i[2] + r_coef_twist*wr_n_i[2]);

j_forces.delta_torque[0] += r_torque[0]*transmittingratio_i + T_torque[0] +
r_coef*wr_t_j[0] + r_coef_twist*wr_n_j[0];

j_forces.delta_torque[1] += r_torque[1]*transmittingratio_i + T_torque[1] +
r_coef*wr_t_j[1] + r_coef_twist*wr_n_j[0];

j_forces.delta_torque[2] += r_torque[2]*transmittingratio_i + T_torque[2] +
r_coef*wr_t_j[2] + r_coef_twist*wr_n_j[0];

/* i_forces.delta_F[0] += r_F[0];

i_forces.delta_F[1] += r_F[1];

i_forces.delta_F[2] += r_F[2];

j_forces.delta_F[0] -= r_F[0];

j_forces.delta_F[1] -= r_F[1];

j_forces.delta_F[2] -= r_F[2];*/

}

void surfacesClose(SurfacesCloseData & scdata, ForceData&, ForceData&)

{

if(scdata.contact_flags) *scdata.contact_flags &=
~CONTACT_ROLLING_MODEL;

```



```

double * const c_history = &scdata.contact_history[history_offset];

c_history[0] = 0.0; // this is the r_torque_old
c_history[1] = 0.0; // this is the r_torque_old
c_history[2] = 0.0; // this is the r_torque_old
c_history[3] = 0;
c_history[4] = 0.0;//kr=0.0;
c_history[5] = 0.0;//kr_O=0.0;
c_history[6] = 0; //firsttouch
c_history[7] = 0.0; // this is the r_torque_xdirection
c_history[8] = 0.0; // this is the r_torque_ydirection
c_history[9] = 0.0; // this is the r_torque_zdirection
c_history[10] = 1.0; //
c_history[11] = 1.0; //
c_history[12] = 0.0; // this is the r_torque_old
c_history[13] = 0.0; // this is the r_torque_old
c_history[14] = 0.0; // this is the r_torque_old
}

void beginPass(SurfacesIntersectData&, ForceData&, ForceData&){ }
void endPass(SurfacesIntersectData&, ForceData&, ForceData&){ }

private:

double ** coeffRollFrict;

double ** coeffFrict;

int history_offset;

    bool treach_flag;

    inline void calcRollTorque(double (&r_torque)[3],double (&T_torque)[3],const
SurfacesIntersectData & sidata,double reff,double wr1,double wr2,double

```

```

wr3,double r_inertia,double (&transmittingratio_i),double (&transmittingratio_j),
double (&r_coef), double (&r_coef_twist)) {

    double wr_n[3],wr_t[3];

    double Ref_T, kr_O;

    const double dt = update->dt;

    double * const c_history = &sidata.contact_history[history_offset]; // requires
Style::TANGENTIAL == TANGENTIAL_HISTORY

    double edges=6;

    double correctionfactor;

    if(sidata.is_wall) {correctionfactor=1;edges*=2;} else {correctionfactor=0.5;}

    const double rmu= coeffRollFrict[sidata.itype][sidata.jtype];

    const double xmu = coeffFrict[sidata.itype][sidata.jtype];

    double Jn = edges/(2*(M_PI-2))*4;

    const double r_torque_max_dash = fabs(sidata.Fn)*reff*rmu;

    double omegamax=100; //mas omega das abgebildet werden soll

    double          kr          =
sidata.kt*reff*reff;//sqrt(kr_O*kr_O/4+M_PI*kr_O/(omegamax*edges*dt))-
kr_O/2;//2*Jn*reff*fabs(sidata.Fn);

    kr_O = fabs(sidata.Fn)*reff*rmu*edges/2;//inclination of the cosine curve
kr*r_torque_max*edges/(kr*M_PI - r_torque_max*edges);

    const          double          r_torque_max          =
M_PI*kr_O/((kr_O/kr+1)*edges);//M_PI*r_torque_max_dash*Jn/(rmu*edges/2
+2*Jn);//most important

    double torque_direction[3];

    double transmittingtorqueratio_i,transmittingtorqueratio_j;

    //starting with a random torque magnitude and direction of torque

```

```
if(c_history[6]==0){
    double normal_vec[3];
    int coni = sidata.i;
    int conj = sidata.j;
    normal_vec[0] = sidata.en[0];
    normal_vec[1] = sidata.en[1];
    normal_vec[2] = sidata.en[2];
    double init_direction[3],directionxyz[3];
    srand((coni+1)*(conj+2));/*time(NULL));
    double perc_init_r_torque = rand() % 101;
    directionxyz[0] = (rand() % 101) - 50;
    directionxyz[1] = (rand() % 101) - 50;
    directionxyz[2] = (rand() % 101) - 50;
    vectorCross3D(normal_vec, directionxyz, init_direction);
    vectorNormalize3D(init_direction);
    double init_r_torque = r_torque_max_dash*perc_init_r_torque/100;
    c_history[0] = init_direction[0]*init_r_torque; // this is the r_torque_old
    c_history[1] = init_direction[1]*init_r_torque; // this is the r_torque_old
    c_history[2] = init_direction[2]*init_r_torque; // this is the r_torque_old
    //c_history[3] = rand() % 2;
    double randx= rand() % 101;
    double randy= rand() % 101;
    //transmittingtorqueratio_j= transmittingtorqueratio_i;//rand() % 101;
    transmittingratio_i=randx/100;
    transmittingratio_j=randy/100;
```

```
transmittingtorqueratio_i=cos(randx/100*M_PI/3);
transmittingtorqueratio_j= 1-transmittingtorqueratio_i;
if(sidata.is_wall) {
    transmittingtorqueratio_i=1;
    transmittingtorqueratio_j=1;
}
c_history[10]=transmittingratio_i;
c_history[11]=transmittingratio_j;
c_history[6]=1;
}
transmittingratio_i = c_history[10];
transmittingratio_j = c_history[11];
const double enx = sidata.en[0];
const double eny = sidata.en[1];
const double enz = sidata.en[2];
//bool treach_flag;
// double maxoverlapc = reff*0.0823922; //=reff*(1/cos(22.5°)-1)
// remove normal (torsion) part of relative rotation
// use only tangential parts for rolling torque
const double wr_dot_delta = wr1*enx+ wr2*eny + wr3*enz; //projection
wr_n[0] = enx * wr_dot_delta;
wr_n[1] = eny * wr_dot_delta;
wr_n[2] = enz * wr_dot_delta;
wr_t[0] = wr1 - wr_n[0];
wr_t[1] = wr2 - wr_n[1];
```

```

wr_t[2] = wr3 - wr_n[2];

// spring (reff depends on wall-particle or particle-particle contact)

double          c_history_mag          =
sqrt(c_history[0]*c_history[0]+c_history[1]*c_history[1]+c_history[2]*c_history
[2]);

//const double r_torque_max = fabs(sidata.Fn)*reff*rmu;//edit + konstant

//    double kr_fit =kr;

double dr_torque[3],dr_F[3];//, wr_tsqu[3];

//    double maxforce=maxoverlapc*sidata.kn/200;//*sidata.deltan;

//    double kF=maxforce*edges/M_PI;

vectorScalarMult3D(wr_t,dt*kr,dr_torque);

double omega_mag = vectorMag3D(wr_t);

    treach_flag=c_history[3]>0.5?1:0;

if(treach_flag==false){

    if(c_history[4]!=0){

        r_torque[0] = c_history[0]*kr/c_history[4] + dr_torque[0];

        r_torque[1] = c_history[1]*kr/c_history[4] + dr_torque[1];

        r_torque[2] = c_history[2]*kr/c_history[4] + dr_torque[2];

    }else{

        r_torque[0] = c_history[0] + dr_torque[0];

        r_torque[1] = c_history[1] + dr_torque[1];

        r_torque[2] = c_history[2] + dr_torque[2];}

    }

// limit max. torque

const double r_torque_mag = vectorMag3D(r_torque);

// const double r_torque_max = fabs(sidata.Fn)*reff*rmu;//edit + konstant

```

```

if(rmu==0){vectorZeroize3D(r_torque);
           c_history[3]=0;}

else{//_____Start of my Rolling Model

double maxtorque_dif;

    if(treach_flag==true){
    if(c_history[5]!=0){
    maxtorque_dif = kr_O/c_history[5];}
    else{maxtorque_dif = 1;}

        vectorScalarMult3D(wr_t,dt*kr_O,dr_torque);

            r_torque[0]      =      c_history[0]*maxtorque_dif      -
dr_torque[0];/*kr_ratio;//

            r_torque[1]      =      c_history[1]*maxtorque_dif      -
dr_torque[1];/*kr_ratio;//

            r_torque[2]      =      c_history[2]*maxtorque_dif      -
dr_torque[2];/*kr_ratio;//

            double directioncheck = c_history[0]*r_torque[0] +
c_history[1]*r_torque[1] + c_history[2]*r_torque[2];

            if(directioncheck<0){

                double randxs= rand() % 101;

                double randys= rand() % 101;

                transmittingratio_i = cos(randxs/100*M_PI/3);
                transmittingratio_j = cos(randys/100*M_PI/3);

                c_history[10]=transmittingratio_i;
                c_history[11]=transmittingratio_j;

            }

        }

else if(r_torque_mag > r_torque_max && treach_flag==0){

```

```

//equation shortened

double kr_O_d_kr_Ox = M_PI/edges*kr_O/r_torque_mag-
kr_O/kr;//M_PI/edges*kr_O/r_torque_mag-edges*rmu/(4*Jn); // always use
R_TORQUE_MAG

r_torque[0] *= kr_O_d_kr_Ox;

r_torque[1] *= kr_O_d_kr_Ox;

r_torque[2] *= kr_O_d_kr_Ox;

c_history[3]=1;
}

//-----

double r_stiff_torque_mag = vectorMag3D(r_torque);

if(r_stiff_torque_mag > r_torque_max && treach_flag == 1)
{
//equation shortened

vectorCopy3D(r_torque, torque_direction);

double kr_d_krx =
M_PI/edges*kr/r_stiff_torque_mag-kr/kr_O;//M_PI/edges*kr/r_stiff_torque_mag-
4*Jn/(edges*rmu); // always use R_STIFFTORQUE_MAG

// if(kr_d_krx>0){

r_torque[0] *= kr_d_krx;

r_torque[1] *= kr_d_krx;

r_torque[2] *= kr_d_krx;

c_history[3]=0;

// }

/* else{

r_torque[0] *= 0;

```

```

        r_torque[1] *= 0;

        r_torque[2] *= 0;

        c_history[3]=0;

    }*/

    //c_history[7] = torque_direction[0];
    c_history[8] = torque_direction[1];
    c_history[9] = torque_direction[2];

    }

    double nabla_torque[3];

    nabla_torque[0] = c_history[0]-r_torque[0];
    nabla_torque[1] = c_history[1]-r_torque[1];
    nabla_torque[2] = c_history[2]-r_torque[2];

// save rolling torque due to spring
c_history[0] = r_torque[0];
c_history[1] = r_torque[1];
c_history[2] = r_torque[2];
c_history[4] = kr;

    c_history[5] = kr_O;

//flattening:

    //transmittingtorqueratio_j = 1-transmittingtorqueratio_i;
    const double torque_mag=vectorMag3D(r_torque);/*
    if(torque_mag>r_torque_max_dash && torque_mag!=0)
    {

        r_torque[0] *= r_torque_max_dash/torque_mag;
        r_torque[1] *= r_torque_max_dash/torque_mag;

```



```

        r_torque[2] *= r_torque_max_dash/torque_mag;
    }
    */
    if(torque_mag<r_torque_max_dash/4 && treach_flag == 1) //keine
    beschleunigungen
    {
        c_history[3]=0;
    }

double acderatio= 1.0;//0.6;//1.0/exp(omega_mag/0.5); //<<---ratio between
accelerating and decelerating torque [0,1]

double rotationflag = r_torque[0]*wr_t[0] + r_torque[1]*wr_t[1] +
r_torque[2]*wr_t[2];

if(rotationflag<0){
    r_torque[0] *= acderatio;
    r_torque[1] *= acderatio;
    r_torque[2] *= acderatio;
}

//damping on each atom seperately see top
    if(c_history[3]==0 && r_torque_max!=0 && torque_mag<r_torque_max){
        r_coef = 1.0 * 2 * sqrt(r_inertia*kr) * (1-torque_mag/r_torque_max);//0.08
        minimal zum ausdämpfen von oszillation zweier partikel
        // add damping torque
//    r_torque[0] += r_coef*wr_t[0];
//    r_torque[1] += r_coef*wr_t[1];
//    r_torque[2] += r_coef*wr_t[2];
        }else{
            r_coef = 0.0 * 2 * sqrt(r_inertia*kr);//

```

```

    // add damping torque
//   r_torque[0] += r_coef*wr_t[0];
//   r_torque[1] += r_coef*wr_t[1];
//   r_torque[2] += r_coef*wr_t[2];
    }

    //instead of damping reduced accelerating torque
}//_____
_____End of my Rolling Model

//torsional part

    const double T_torque_max = fabs(sidata.Fn)*reff*xmu*0.65;

    const double krT = sidata.kt*reff*reff/2;//T_torque_max/(omegamax*dt);//
/M_PI*90;

    double dr_torqueT[3];

    vectorScalarMult3D(wr_n,dt*krT,dr_torqueT);

    if(c_history[7]!=0){

        T_torque[0] = c_history[12]*krT/c_history[7] +
dr_torqueT[0];

        T_torque[1] = c_history[13]*krT/c_history[7] +
dr_torqueT[1];

        T_torque[2] = c_history[14]*krT/c_history[7] +
dr_torqueT[2];

    }else{

        T_torque[0] = c_history[12] + dr_torqueT[0];

        T_torque[1] = c_history[13] + dr_torqueT[1];

        T_torque[2] = c_history[14] + dr_torqueT[2];}

// limit max. torque

    const double T_torque_mag = vectorMag3D(T_torque);

```

```

if(T_torque_mag > T_torque_max)
{
    //printf("[%d]   %e   >   %e\n",   update->ntimestep,   r_torque_mag,
r_torque_max);

    const double factor = T_torque_max / T_torque_mag;

    T_torque[0] *= factor;
    T_torque[1] *= factor;
    T_torque[2] *= factor;

    c_history[12] = T_torque[0];
    c_history[13] = T_torque[1];
    c_history[14] = T_torque[2];

    c_history[7] = krT;

    r_coef_twist=0;

    }else{

    // save rolling torque due to spring

    c_history[12] = T_torque[0];
    c_history[13] = T_torque[1];
    c_history[14] = T_torque[2];

    c_history[7] = krT;

        r_coef_twist=0;

        if(T_torque_max!=0){

            r_coef_twist   =   0.3   *   2   *   sqrt(r_inertia*kr)   *   (1-
T_torque_mag/T_torque_max);//minimal zum ausdämpfen von oszillation zweier
partikel

            // add damping torque

            //   T_torque[0] += r_coef_twist*wr_n[0];

            //   T_torque[1] += r_coef_twist*wr_n[1];

```

```

//   T_torque[2] += r_coef_twist*wr_n[2];
}

// no damping / no dashpot in case of full mobilisation rolling angle
}

/*

const double T_twisttorque_delta = r_torque[0]*enx+ r_torque[1]*eny +
r_torque[2]*enz; //projection

    r_torque[0] -= enx * T_twisttorque_delta;
    r_torque[1] -= eny * T_twisttorque_delta;
    r_torque[2] -= enz * T_twisttorque_delta;*/
}

};

}

}

#endif // ROLLING_MODEL_EPSD_H_

#endif

```

8.4 Modified rolling model dahl2

```

#ifndef ROLLING_MODEL
ROLLING_MODEL(ROLLING_D AHL2,dahl2,8)
#else
#ifndef ROLLING_MODEL_D AHL2_H_
#define ROLLING_MODEL_D AHL2_H_
#include "contact_models.h"
#include <algorithm>
#include "math.h"

```

```
#include "domain.h"

#include "math_extra_liggghts.h"

namespace LIGGGHTS {
namespace ContactModels
{
using namespace LAMMPS_NS;

template<>
class RollingModel<ROLLING_D AHL2> : protected Pointers
{
public:
    static const int MASK = CM_CONNECT_TO_PROPERTIES |
CM_SURFACES_INTERSECT | CM_SURFACES_CLOSE;

    RollingModel(class LAMMPS * Imp, IContactHistorySetup * hsetup, class
ContactModelBase *) :
        Pointers(Imp), coeffRollFrict(NULL)
    {
        history_offset = hsetup->add_history_value("rollangle", "1");
        hsetup->add_history_value("rollangle", "1");
        hsetup->add_history_value("rollanglez", "1");
        hsetup->add_history_value("plastic_angle", "1");
        hsetup->add_history_value("plastic_angle", "1");
        hsetup->add_history_value("plastic_anglez", "1");
        hsetup->add_history_value("stored_angle", "1");
        hsetup->add_history_value("stored_angle", "1");
        hsetup->add_history_value("stored_anglez", "1");
        hsetup->add_history_value("delta_angle", "1");
    }
};
}
```

```
hsetup->add_history_value("delta_angley", "1");
hsetup->add_history_value("delta_anglez", "1");
hsetup->add_history_value("T_rollanglex", "1");
hsetup->add_history_value("T_rollangley", "1");
hsetup->add_history_value("T_rollanglez", "1");
hsetup->add_history_value("T_plastic_anglex", "1");
hsetup->add_history_value("T_plastic_angley", "1");
hsetup->add_history_value("T_plastic_anglez", "1");
hsetup->add_history_value("T_stored_anglex", "1");
hsetup->add_history_value("T_stored_angley", "1");
hsetup->add_history_value("T_stored_anglez", "1");
hsetup->add_history_value("T_delta_anglex", "1");
hsetup->add_history_value("T_delta_angley", "1");
hsetup->add_history_value("T_delta_anglez", "1");
hsetup->add_history_value("free", "0");
hsetup->add_history_value("losbrechmoment", "0");
}

void registerSettings(Settings&) {}

void connectToProperties(PropertyRegistry & registry) {

    registry.registerProperty("coeffRollFrict",
&MODEL_PARAMS::createCoeffRollFrict);

    registry.connect("coeffRollFrict", coeffRollFrict, "rolling_model dahl2");

    registry.registerProperty("coeffFrict",
&MODEL_PARAMS::createCoeffFrict);

    registry.connect("coeffFrict", coeffFrict, "rolling_model dahl2");

    // error checks on coarsegraining
```

```

    if(force->cg_active())
        error->cg(FLERR,"rolling model dahl2");
    }

    void surfacesIntersect(SurfacesIntersectData & sidata, ForceData & i_forces,
    ForceData & j_forces)
    {
        double
r_torque[3],T_torque[3],r_coef,wr_n_i[3],wr_n_j[3],wr_t_i[3],wr_t_j[3];

        vectorZeroize3D(r_torque);

        vectorZeroize3D(T_torque);

        if(sidata.contact_flags == CONTACT_ROLLING_MODEL;

            const double radi = sidata.radi;

            const double radj = sidata.radj;

            double reff=sidata.is_wall ? sidata.radi : (radi*radj/(radi+radj));

#ifdef SUPERQUADRIC_ACTIVE_FLAG

            if(sidata.is_non_spherical)

                reff = MathExtraLiggghtsSuperquadric::get_effective_radius(sidata);

#endif

            if(sidata.is_wall) {

                const double wr1 = sidata.wr1;

                const double wr2 = sidata.wr2;

                const double wr3 = sidata.wr3;

                const double radius = sidata.radi;

                double r_inertia;

                if (domain->dimension == 2) r_inertia = 1.5*sidata.mi*radius*radius;

                else r_inertia = 1.4*sidata.mi*radius*radius;

```

```

    calcRollTorque(r_torque,T_torque,sidata,reff,wr1,wr2,wr3,r_inertia,r_coef);
/*
    const double wr_dot_delta_i = sidata.en[0]*wr1 + sidata.en[1]*wr2 +
sidata.en[2]*wr3; //projection

    vectorScalarMult3D(sidata.en, wr_dot_delta_i, wr_n_i);

    wr_t_i[0]=wr1 -wr_n_i[0];

    wr_t_i[1]=wr2 -wr_n_i[1];

    wr_t_i[2]=wr3 -wr_n_i[2];

    vectorCopy3D(wr_n_i, wr_n_j);

    vectorCopy3D(wr_t_i, wr_t_j);*/
} else {

    double wr_roll[3];

    const int i = sidata.i;

    const int j = sidata.j;

    const double * const * const omega = atom->omega;

    const double r_inertia_red_i = sidata.mi*radi*radi;

    const double r_inertia_red_j = sidata.mj*radj*radj;

    double r_inertia;

    if (domain->dimension == 2) r_inertia = 1.5 * r_inertia_red_i *
r_inertia_red_j/(r_inertia_red_i + r_inertia_red_j);

    else r_inertia = 1.4 * r_inertia_red_i * r_inertia_red_j/(r_inertia_red_i +
r_inertia_red_j);

    // relative rotational velocity

    vectorSubtract3D(omega[i],omega[j],wr_roll);

calcRollTorque(r_torque,T_torque,sidata,reff,wr_roll[0],wr_roll[1],wr_roll[2],r_i
nertia,r_coef);

```



```

/*
    const double wr_dot_delta_i = vectorDot3D(omega[i],sidata.en);
//projection
    vectorScalarMult3D(sidata.en, wr_dot_delta_i, wr_n_i);
    vectorSubtract3D(omega[i],wr_n_i, wr_t_i);
    const double wr_dot_delta_j = vectorDot3D(omega[j],sidata.en);
//projection
    vectorScalarMult3D(sidata.en, wr_dot_delta_j, wr_n_j);
    vectorSubtract3D(omega[j],wr_n_j, wr_t_j);*/
}

i_forces.delta_torque[0] -= (r_torque[0]+T_torque[0]);//+r_coef*wr_t_i[0];
i_forces.delta_torque[1] -= (r_torque[1]+T_torque[1]);//+r_coef*wr_t_i[1];
i_forces.delta_torque[2] -= (r_torque[2]+T_torque[2]);//+r_coef*wr_t_i[2];
j_forces.delta_torque[0] += r_torque[0]+T_torque[0];//+r_coef*wr_t_j[0];
j_forces.delta_torque[1] += r_torque[1]+T_torque[1];//+r_coef*wr_t_j[1];
j_forces.delta_torque[2] += r_torque[2]+T_torque[2];//+r_coef*wr_t_j[2];
}

void surfacesClose(SurfacesCloseData & scdata, ForceData&, ForceData&)
{
    if(scdata.contact_flags)                *scdata.contact_flags                &=
~CONTACT_ROLLING_MODEL;

    double * const c_history = &scdata.contact_history[history_offset];

    c_history[0] = 0.0; // dangle
    c_history[1] = 0.0; // dangle
    c_history[2] = 0.0; // dangle
    c_history[3] = 0.0; //
    c_history[4] = 0.0; //

```

```
c_history[5] = 0.0; //
c_history[6] = 0.0; //
c_history[7] = 0.0; //
c_history[8] = 0.0; //
c_history[9] = 0.0;
c_history[10] = 0.0;
c_history[11] = 0.0;
c_history[12] = 0.0; // Tdangle
c_history[13] = 0.0; // Tdangle
c_history[14] = 0.0; // Tdangle
c_history[15] = 0.0; //
c_history[16] = 0.0; //
c_history[17] = 0.0; //
c_history[18] = 0.0; //
c_history[19] = 0.0; //
c_history[20] = 0.0; //
c_history[21] = 0.0;
c_history[22] = 0.0;
c_history[23] = 0.0;
c_history[24] = 0;
c_history[25] = 0;
}

void beginPass(SurfacesIntersectData&, ForceData&, ForceData&){}
void endPass(SurfacesIntersectData&, ForceData&, ForceData&){}

private:
```

```

double ** coeffRollFrict;

double ** coeffFrict;

int history_offset;

inline void calcRollTorque(double (&r_torque)[3],double (&T_torque)[3],const
SurfacesIntersectData & sidata,double reff,double wr1,double wr2,double wr3,
double r_inertia, double (&r_coef)) {

    double wr_n[3],wr_t[3];

    double losbrechmoment; //110% = 1.1

    double old_roll[3],d_old_roll[3];

    const double enx = sidata.en[0];

    const double eny = sidata.en[1];

    const double enz = sidata.en[2];

    double tangentialdirection[3];

    const double dt = update->dt;

    double * const c_history = &sidata.contact_history[history_offset]; // requires
Style::TANGENTIAL == TANGENTIAL_HISTORY

    const double rmu = coeffRollFrict[sidata.itype][sidata.jtype];

    const double xmu = coeffFrict[sidata.itype][sidata.jtype];

    bool nulldurchgang=0;

        losbrechmoment=1.0;

    // remove normal (torsion) part of relative rotation

    // use only tangential parts for rolling torque

    const double wr_dot_delta = wr1*enx+ wr2*eny + wr3*enz;

    wr_n[0] = enx * wr_dot_delta;

    wr_n[1] = eny * wr_dot_delta;

    wr_n[2] = enz * wr_dot_delta;

    wr_t[0] = wr1 - wr_n[0];

```

```

wr_t[1] = wr2 - wr_n[1];

wr_t[2] = wr3 - wr_n[2];

// spring (reff depends on wall-particle or particle-particle contact)

const double kr = sidata.kt*reff*reff *2;
                //<<<<<<<<<<<<<<<-----EDIT

r_coef = 2.0*sqrt(r_inertia*kr);

double dr_angle[3];

vectorScalarMult3D(wr_t,dt,dr_angle);

//rotate the rolling

                double                quad_oldrollmag                =
sqrt(c_history[0]*c_history[0]+c_history[1]*c_history[1]+c_history[2]*c_history
[2]);

                double normalpart = c_history[0]*enx + c_history[1]*eny +
c_history[2]*enz;

tangentialdirection[0] = c_history[0] - normalpart * enx;

tangentialdirection[1] = c_history[1] - normalpart * eny;

tangentialdirection[2] = c_history[2] - normalpart * enz;

vectorNormalize3D(tangentialdirection);

                c_history[0] = quad_oldrollmag * tangentialdirection[0];

                c_history[1] = quad_oldrollmag * tangentialdirection[1];

                c_history[2] = quad_oldrollmag * tangentialdirection[2];

                quad_oldrollmag                =
sqrt(c_history[3]*c_history[3]+c_history[4]*c_history[4]+c_history[5]*c_history
[5]);

                normalpart = c_history[3]*enx + c_history[4]*eny +
c_history[5]*enz;

tangentialdirection[0] = c_history[3] - normalpart * enx;

```

```

tangentialdirection[1] = c_history[4] - normalpart * eny;
tangentialdirection[2] = c_history[5] - normalpart * enz;
vectorNormalize3D(tangentialdirection);

    c_history[3] = quad_oldrollmag * tangentialdirection[0];
    c_history[4] = quad_oldrollmag * tangentialdirection[1];
    c_history[5] = quad_oldrollmag * tangentialdirection[2];

    quad_oldrollmag
sqrt(c_history[6]*c_history[6]+c_history[7]*c_history[7]+c_history[8]*c_history
[8]);
    normalpart = c_history[6]*enx + c_history[7]*eny +
c_history[8]*enz;

tangentialdirection[0] = c_history[6] - normalpart * enx;
tangentialdirection[1] = c_history[7] - normalpart * eny;
tangentialdirection[2] = c_history[8] - normalpart * enz;
vectorNormalize3D(tangentialdirection);

    c_history[6] = quad_oldrollmag * tangentialdirection[0];
    c_history[7] = quad_oldrollmag * tangentialdirection[1];
    c_history[8] = quad_oldrollmag * tangentialdirection[2];

    quad_oldrollmag
sqrt(c_history[9]*c_history[9]+c_history[10]*c_history[10]+c_history[11]*c_his
tory[11]);
    normalpart = c_history[9]*enx + c_history[10]*eny +
c_history[11]*enz;

tangentialdirection[0] = c_history[9] - normalpart * enx;
tangentialdirection[1] = c_history[10] - normalpart * eny;
tangentialdirection[2] = c_history[11] - normalpart * enz;
vectorNormalize3D(tangentialdirection);

    c_history[9] = quad_oldrollmag * tangentialdirection[0];

```

```

c_history[10] = quad_oldrollmag * tangentialdirection[1];
c_history[11] = quad_oldrollmag * tangentialdirection[2];
##### end of rotations
d_old_roll[0] = c_history[0];
d_old_roll[1] = c_history[1];
d_old_roll[2] = c_history[2];
c_history[0] += dr_angle[0];
c_history[1] += dr_angle[1];
c_history[2] += dr_angle[2];
old_roll[0] = c_history[3];
old_roll[1] = c_history[4];
old_roll[2] = c_history[5];
c_history[3] += dr_angle[0];
c_history[4] += dr_angle[1];
c_history[5] += dr_angle[2];
c_history[9] += dr_angle[0];
c_history[10] += dr_angle[1];
c_history[11] += dr_angle[2];

const double rollmagat =
sqrt(c_history[0]*c_history[0]+c_history[1]*c_history[1]+c_history[2]*c_history
[2]);

double magnitde =
sqrt(c_history[3]*c_history[3]+c_history[4]*c_history[4]+c_history[5]*c_history
[5]);

const double
compnulldurchgang=c_history[0]*old_roll[0]+c_history[1]*old_roll[1]+c_histor
y[2]*old_roll[2];

```

```

    if(compnulldurchgang<0||((c_history[0]*d_old_roll[0]+c_history[1]*d_old_
    _roll[1]+c_history[2]*d_old_roll[2])<0)){

        nulldurchgang=1;

    }

    if(nulldurchgang==1){

        double buff_roll[3];

        buff_roll[0] = old_roll[0]-d_old_roll[0];

        buff_roll[1] = old_roll[1]-d_old_roll[1];

        buff_roll[2] = old_roll[2]-d_old_roll[2];

        c_history[3] = c_history[9] + c_history[6];

        c_history[4] = c_history[10] + c_history[7];

        c_history[5] = c_history[11] + c_history[8];

        if((buff_roll[0]*c_history[3]+buff_roll[1]*c_history[4]+buff_roll[2]*c_his
        tory[5])>0){

            c_history[3] = c_history[0];

            c_history[4] = c_history[1];

            c_history[5] = c_history[2];

        }

        c_history[6] = buff_roll[0];

        c_history[7] = buff_roll[1];

        c_history[8] = buff_roll[2];

        c_history[9] = c_history[0];

        c_history[10] = c_history[1];

        c_history[11] = c_history[2];

    }

    double  rsht  =  c_history[0]*enx  +  c_history[1]*eny  +
    c_history[2]*enz;

```

```

tangentialdirection[0] = c_history[0] - rsht * enx;
tangentialdirection[1] = c_history[1] - rsht * eny;
tangentialdirection[2] = c_history[2] - rsht * enz;
vectorNormalize3D(tangentialdirection);

const double rollmag =
sqrt(c_history[3]*c_history[3]+c_history[4]*c_history[4]+c_history[5]*c_history
[5]);

//unloading/reloading
r_torque[0] = kr*c_history[0];
r_torque[1] = kr*c_history[1];
r_torque[2] = kr*c_history[2];

double r_torque_max;
if((kr/2*rollmag)>(losbrechmoment*fabs(sidata.Fn)*reff*rmu)){
    r_torque_max = fabs(sidata.Fn)*reff*rmu;
    c_history[3] *= r_torque_max/kr*2 /rollmag;
    c_history[4] *= r_torque_max/kr*2 /rollmag;
    c_history[5] *= r_torque_max/kr*2 /rollmag;
}else{
    r_torque_max = kr/2*rollmag;
}

const double r_torque_mag = vectorMag3D(r_torque);
if(r_torque_mag > r_torque_max)
{
    if(r_torque_mag != 0.0 && kr!=0){

```



```

const double factor = r_torque_max / r_torque_mag;

r_torque[0] *= factor;

r_torque[1] *= factor;

r_torque[2] *= factor;

// save rolling torque due to spring

c_history[0] = r_torque[0]/kr;

c_history[1] = r_torque[1]/kr;

c_history[2] = r_torque[2]/kr;

    }else{

        vectorZeroize3D(r_torque);

        //c_history[0]=c_history[1]=c_history[2]=0;

    }

}else{

    r_coef = 2.0*sqrt(r_inertia*kr);

}

r_torque[0] += r_coef*wr_t[0];

r_torque[1] += r_coef*wr_t[1];

r_torque[2] += r_coef*wr_t[2];

const double r_mag_withdamp = vectorMag3D(r_torque);

//double          signofdirection          =
((r_torque[0]*tangentialdirection[0]+r_torque[1]*tangentialdirection[1]+r_torque
[2]*tangentialdirection[2])>0) ? 1 : -1;

const double Fr_coulomb = fabs(sidata.Fn)*reff*rmu;

if (r_mag_withdamp > losbrechmoment*Fr_coulomb) {

    const double ratiod = Fr_coulomb / r_mag_withdamp;

```

```

    r_torque[0] *= ratioid;

    r_torque[1] *= ratioid;

    r_torque[2] *= ratioid;

}

//=====TORSIONAL PART

    bool nulldurchgangtwist;

    double normaldirection[3];

    double r_coef_twist;

    //double item = 1.0-sidata.deltan/(2*reff);

    const double T_torque_max_Coulomb =
fabs(sidata.Fn)*xmu*0.65*reff*rmu;//sqrt(1.0-item*item);

    const double krT = sidata.kt*reff*reff;

    double dr_torqueT[3];

    double dt_angle[3];

    vectorScalarMult3D(wr_n,dt,dt_angle);

        //rotate the rolling

        double quad_oldtwistmag =
sqrt(c_history[12]*c_history[12]+c_history[13]*c_history[13]+c_history[14]*c_
history[14]);

        normalpart = c_history[12]*enx + c_history[13]*eny +
c_history[14]*enz;

    normaldirection[0] = normalpart * enx;

    normaldirection[1] = normalpart * eny;

    normaldirection[2] = normalpart * enz;

    vectorNormalize3D(normaldirection);

        c_history[12] = quad_oldtwistmag * normaldirection[0];

        c_history[13] = quad_oldtwistmag * normaldirection[1];

```

```

c_history[14] = quad_oldtwistmag * normaldirection[2];

quad_oldtwistmag =
sqrt(c_history[15]*c_history[15]+c_history[16]*c_history[16]+c_history[17]*c_
history[17]);

normalpart = c_history[15]*enx + c_history[16]*eny +
c_history[17]*enz;

normaldirection[0] = normalpart * enx;
normaldirection[1] = normalpart * eny;
normaldirection[2] = normalpart * enz;
vectorNormalize3D(normaldirection);

c_history[15] = quad_oldtwistmag * normaldirection[0];
c_history[16] = quad_oldtwistmag * normaldirection[1];
c_history[17] = quad_oldtwistmag * normaldirection[2];

quad_oldtwistmag =
sqrt(c_history[18]*c_history[18]+c_history[19]*c_history[19]+c_history[20]*c_
history[20]);

normalpart = c_history[18]*enx + c_history[19]*eny +
c_history[20]*enz;

normaldirection[0] = normalpart * enx;
normaldirection[1] = normalpart * eny;
normaldirection[2] = normalpart * enz;
vectorNormalize3D(normaldirection);

c_history[18] = quad_oldtwistmag * normaldirection[0];
c_history[19] = quad_oldtwistmag * normaldirection[1];
c_history[20] = quad_oldtwistmag * normaldirection[2];

quad_oldtwistmag =
sqrt(c_history[21]*c_history[21]+c_history[22]*c_history[22]+c_history[23]*c_
history[23]);

```

```

        normalpart = c_history[21]*enx + c_history[22]*eny +
c_history[23]*enz;

    normaldirection[0] = normalpart * enx;

    normaldirection[1] = normalpart * eny;

    normaldirection[2] = normalpart * enz;

    vectorNormalize3D(normaldirection);

        c_history[21] = quad_oldtwistmag * normaldirection[0];

        c_history[22] = quad_oldtwistmag * normaldirection[1];

        c_history[23] = quad_oldtwistmag * normaldirection[2];

        c_history[12] += dt_angle[0];

        c_history[13] += dt_angle[1];

        c_history[14] += dt_angle[2];

        const          double          twistmagat          =
sqrt(c_history[12]*c_history[12]+c_history[13]*c_history[13]+c_history[14]*c_
history[14]);

        double          magnitdetwist          =
sqrt(c_history[15]*c_history[15]+c_history[16]*c_history[16]+c_history[17]*c_
history[17]);

        const          double
compnulldurchgangT=c_history[12]*c_history[15]+c_history[13]*c_history[16]
+c_history[14]*c_history[17];

        if(compnulldurchgangT<0){//(magnitde==0.0          &&
vectorDot3D(oldroll,dr_angle)<0){

            nulldurchgangtwist=1;

        }

        if(nulldurchgangtwist==1){

            double buff_twist[3];

            buff_twist[0] = c_history[15];

            buff_twist[1] = c_history[16];

```

```

buff_twist[2] = c_history[17];

c_history[15] = c_history[21] + c_history[18];

c_history[16] = c_history[22] + c_history[19];

c_history[17] = c_history[23] + c_history[20];

if((buff_twist[0]*c_history[15]+buff_twist[1]*c_history[16]+buff_twist[2
]*c_history[17])>0){

    c_history[15] = c_history[12];

    c_history[16] = c_history[13];

    c_history[17] = c_history[14];

    }

    c_history[18] = buff_twist[0];

    c_history[19] = buff_twist[1];

    c_history[20] = buff_twist[2];

    c_history[21] = 0.0;

    c_history[22] = 0.0;

    c_history[23] = 0.0;

}

c_history[15] += dr_angle[0];

c_history[16] += dr_angle[1];

c_history[17] += dr_angle[2];

c_history[21] += dr_angle[0];

c_history[22] += dr_angle[1];

c_history[23] += dr_angle[2];

double  rshtT  =  c_history[12]*enx  +  c_history[13]*eny  +
c_history[14]*enz;

normaldirection[0] = rshtT * enx;

```

```

normaldirection[1] = rshtT * eny;

normaldirection[2] = rshtT * enz;

vectorNormalize3D(normaldirection);

const double twistmag =
sqrt(c_history[15]*c_history[15]+c_history[16]*c_history[16]+c_history[17]*c_
history[17]);

T_torque[0] = krT*c_history[12];

T_torque[1] = krT*c_history[13];

T_torque[2] = krT*c_history[14];

double T_torque_max;//T_torque_max_Coulomb

if((krT/2*twistmag)>(losbrechmoment*T_torque_max_Coulomb)){

    T_torque_max = fabs(sidata.Fn)*reff*rmu;

    c_history[3] *= T_torque_max/krT*2 /twistmag;

    c_history[4] *= T_torque_max/krT*2 /twistmag;

    c_history[5] *= T_torque_max/krT*2 /twistmag;

    }else{

        T_torque_max = krT/2*twistmag;

    }

const double T_torque_mag = vectorMag3D(T_torque);

if(T_torque_mag > T_torque_max)

{

    if(T_torque_mag != 0.0){

const double Tfactor = T_torque_max / T_torque_mag;

T_torque[0] *= Tfactor;

T_torque[1] *= Tfactor;

T_torque[2] *= Tfactor;

```

```

c_history[12] = T_torque[0]/krT;
c_history[13] = T_torque[1]/krT;
c_history[14] = T_torque[2]/krT;
        }else{T_torque[0]=T_torque[1]=T_torque[2]=0.0;}
    }else{
r_coef_twist = 0.3* 2.0 * sqrt(r_inertia*krT);
    }

        r_coef_twist = 0.3* 2.0 * sqrt(r_inertia*krT);
        T_torque[0] += r_coef_twist*wr_n[0];
T_torque[1] += r_coef_twist*wr_n[1];
T_torque[2] += r_coef_twist*wr_n[2];
//=====end of torsional part
const double c_T_torque_mag = vectorMag3D(T_torque);
if(c_T_torque_mag > losbrechmoment*T_torque_max_Coulomb)
{
        double          factorialT          =
T_torque_max_Coulomb/c_T_torque_mag;
        vectorScalarMult3D(T_torque, factorialT);
    }
    #####
}
};
}
}
#endif // ROLLING_MODEL_D AHL2_H_
#endif

```

8.5 Modified tangential model dahl

```
#ifndef TANGENTIAL_MODEL

TANGENTIAL_MODEL(TANGENTIAL_D AHL,dahl,4)

#else

#ifndef TANGENTIAL_MODEL_D AHL_H_

#define TANGENTIAL_MODEL_D AHL_H_

#include "contact_models.h"

#include "math.h"

#include "update.h"

#include "global_properties.h"

#include "atom.h"

namespace LIGGGHTS {

namespace ContactModels

{

template<>

class TangentialModel<TANGENTIAL_D AHL> : protected Pointers

{

double ** coeffFrict;

int history_offset;

public:

static const int MASK = CM_CONNECT_TO_PROPERTIES |

CM_SURFACES_INTERSECT | CM_SURFACES_CLOSE;

TangentialModel(LAMMPS * Imp, IContactHistorySetup * hsetup,class

ContactModelBase *c) : Pointers(Imp),

coeffFrict(NULL),

heating(false),
```



```
    heating_track(false),
    cmb(c)
{
    history_offset = hsetup->add_history_value("shearx", "1");
    hsetup->add_history_value("sheary", "1");
    hsetup->add_history_value("shearz", "1");
    hsetup->add_history_value("plastic_shearx", "1");
    hsetup->add_history_value("plastic_sheary", "1");
    hsetup->add_history_value("plastic_shearz", "1");
    hsetup->add_history_value("stored_shearx", "1");
    hsetup->add_history_value("stored_sheary", "1");
    hsetup->add_history_value("stored_shearz", "1");
    hsetup->add_history_value("delta_shearx", "1");
    hsetup->add_history_value("delta_sheary", "1");
    hsetup->add_history_value("delta_shearz", "1");
    hsetup->add_history_value("free", "0");
    hsetup->add_history_value("losbrechmoment", "0");
}

inline void registerSettings(Settings& settings)
{
    settings.registerOnOff("heating_tangential_dahl",heating,false);
    settings.registerOnOff("heating_tracking",heating_track,false);

    //TODO error->one(FLERR,"TODO here also check if right surface model
used");
}

inline void connectToProperties(PropertyRegistry & registry)
```

```

{
    registry.registerProperty("coeffFrict",
&MODEL_PARAMS::createCoeffFrict);

    registry.connect("coeffFrict", coeffFrict, "tangential_model dahl");
}

inline void surfacesIntersect(const SurfacesIntersectData & sidata, ForceData &
i_forces, ForceData & j_forces)
{
    // normal forces = Hookian contact + normal velocity damping

    const double enx = sidata.en[0];
    const double eny = sidata.en[1];
    const double enz = sidata.en[2];
    double tangentialdirection[3];
    bool nulldurchgang=0;
    double losbrechmoment; //110% = 1.1
    double old_shear[3],dx_old_shear[3];

    // shear history effects

    if(sidata.contact_flags) *sidata.contact_flags |=
CONTACT_TANGENTIAL_MODEL;

    double * const shear = &sidata.contact_history[history_offset];

        const double dt = update->dt;

        losbrechmoment=1.0;

    if (sidata.shearupdate && sidata.computeflag) {

        // rotate shear displacements //normal part entfernen

        double shrmagbefore = sqrt(shear[0]*shear[0] + shear[1]*shear[1] +
shear[2]*shear[2]);

```

```

    double normalpart = shear[0]*enx + shear[1]*eny + shear[2]*enz;
    tangentialdirection[0] = shear[0] - normalpart * enx;
    tangentialdirection[1] = shear[1] - normalpart * eny;
    tangentialdirection[2] = shear[2] - normalpart * enz;
    vectorNormalize3D(tangentialdirection);
    shear[0] = shrmagbefore * tangentialdirection[0];
    shear[1] = shrmagbefore * tangentialdirection[1];
    shear[2] = shrmagbefore * tangentialdirection[2];
    shrmagbefore = sqrt(shear[3]*shear[3] + shear[4]*shear[4] +
shear[5]*shear[5]);
    normalpart = shear[3]*enx + shear[4]*eny + shear[5]*enz;
    tangentialdirection[0] = shear[3] - normalpart * enx;
    tangentialdirection[1] = shear[4] - normalpart * eny;
    tangentialdirection[2] = shear[5] - normalpart * enz;
    vectorNormalize3D(tangentialdirection);
    shear[3] = shrmagbefore * tangentialdirection[0];
    shear[4] = shrmagbefore * tangentialdirection[1];
    shear[5] = shrmagbefore * tangentialdirection[2];
    shrmagbefore = sqrt(shear[6]*shear[6] + shear[7]*shear[7] +
shear[8]*shear[8]);
    normalpart = shear[6]*enx + shear[7]*eny + shear[8]*enz;
    tangentialdirection[0] = shear[6] - normalpart * enx;
    tangentialdirection[1] = shear[7] - normalpart * eny;
    tangentialdirection[2] = shear[8] - normalpart * enz;
    vectorNormalize3D(tangentialdirection);
    shear[6] = shrmagbefore * tangentialdirection[0];

```

```

shear[7] = shrmagbefore * tangentialdirection[1];

shear[8] = shrmagbefore * tangentialdirection[2];

shrmagbefore = sqrt(shear[9]*shear[9] + shear[10]*shear[10] +
shear[11]*shear[11]);

normalpart = shear[9]*enx + shear[10]*eny + shear[11]*enz;

tangentialdirection[0] = shear[9] - normalpart * enx;

tangentialdirection[1] = shear[10] - normalpart * eny;

tangentialdirection[2] = shear[11] - normalpart * enz;

vectorNormalize3D(tangentialdirection);

shear[9] = shrmagbefore * tangentialdirection[0];

shear[10] = shrmagbefore * tangentialdirection[1];

shear[11] = shrmagbefore * tangentialdirection[2];

#####end of rotation

dx_old_shear[0] = shear[0];

dx_old_shear[1] = shear[1];

dx_old_shear[2] = shear[2];

shear[0] += sidata.vtr1 * dt;

shear[1] += sidata.vtr2 * dt;

shear[2] += sidata.vtr3 * dt;

old_shear[0] = shear[3];

old_shear[1] = shear[4];

old_shear[2] = shear[5];

shear[3] += sidata.vtr1 * dt;

shear[4] += sidata.vtr2 * dt;

shear[5] += sidata.vtr3 * dt;

shear[9] += sidata.vtr1 * dt;

```

```

shear[10] += sidata.vtr2 * dt;

shear[11] += sidata.vtr3 * dt;

}

const double shrmag = sqrt(shear[0]*shear[0] + shear[1]*shear[1] +
shear[2]*shear[2]);

const double kt = sidata.kt *2.0;
                /*pow(fabs(sidata.Fn),2.0/3.0);

const double xmu = coeffFrict[sidata.itype][sidata.jtype];

const double magnitde = sqrt(shear[3]*shear[3] + shear[4]*shear[4] +
shear[5]*shear[5]);

const double compnulldurchgang=shear[0]*shear[3]+shear[1]*shear[4]+shear[2]*shear[5];//sh
ear[0]*old_shear[0]+shear[1]*old_shear[1]+shear[2]*old_shear[2];

if(compnulldurchgang<0/*||(shear[0]*dx_old_shear[0]+shear[1]*dx_old_s
hear[1]+shear[2]*dx_old_shear[2])<0*/){

    nulldurchgang=1;

}

if(nulldurchgang==1){

    double buff_shear[3];

    buff_shear[0] = old_shear[0]-dx_old_shear[0];

    buff_shear[1] = old_shear[1]-dx_old_shear[1];

    buff_shear[2] = old_shear[2]-dx_old_shear[2];

    shear[3] = shear[9] + shear[6];

    shear[4] = shear[10] + shear[7];

    shear[5] = shear[11] + shear[8];

    if((buff_shear[0]*shear[3]+buff_shear[1]*shear[4]+buff_shear[2]*shear[5
])>0){

        shear[3] = shear[0];

```

```

        shear[4] = shear[1];
        shear[5] = shear[2];
    }
    shear[6] = buff_shear[0];
    shear[7] = buff_shear[1];
    shear[8] = buff_shear[2];
    shear[9] = shear[0];
    shear[10] = shear[1];
    shear[11] = shear[2];
}

double rsht = shear[0]*enx + shear[1]*eny + shear[2]*enz;
tangentialdirection[0] = shear[0] - rsht * enx;
tangentialdirection[1] = shear[1] - rsht * eny;
tangentialdirection[2] = shear[2] - rsht * enz;
vectorNormalize3D(tangentialdirection);

const double shearedmag = sqrt(shear[3]*shear[3] + shear[4]*shear[4] +
shear[5]*shear[5]);

    const double gammat = sidata.gammat;

// tangential forces = shear
double Ft1 = -(kt * shear[0]);
double Ft2 = -(kt * shear[1]);
double Ft3 = -(kt * shear[2]);

    //yield algorithm
const double Ft_shear = sqrt(Ft1 * Ft1 + Ft2 * Ft2 + Ft3 * Ft3);
double Ft_friction;
if((kt/2*shearedmag)>(losbrechmoment*xmu*fabs(sidata.Fn))){

```

```

    Ft_friction = xmu * fabs(sidata.Fn);

    shear[3] *= Ft_friction/kt*2 /shearedmag;

    shear[4] *= Ft_friction/kt*2 /shearedmag;

    shear[5] *= Ft_friction/kt*2 /shearedmag;

    }else{

        Ft_friction = kt/2*shearedmag;

    }

//plastic shear

if (Ft_shear > Ft_friction) {

    if (shrmag != 0.0 && kt != 0.0) {

        const double ratio = Ft_friction / Ft_shear;

        if(heating)

        {

            sidata.P_diss      +=      (vectorMag3DSquared(shear)*kt      -
ratio*ratio*vectorMag3DSquared(shear)*kt) / (update->dt);

            if(heating_track      &&      sidata.is_wall)      cmb-
>tally_pw((vectorMag3DSquared(shear)*kt      -
ratio*ratio*vectorMag3DSquared(shear)*kt)      /      (update-
>dt),sidata.i,sidata.jtype,2);

            if(heating_track      &&      !sidata.is_wall)      cmb-
>tally_pp((vectorMag3DSquared(shear)*kt      -
ratio*ratio*vectorMag3DSquared(shear)*kt) / (update->dt),sidata.i,sidata.j,2);

        }

        Ft1 *= ratio;

        Ft2 *= ratio;

        Ft3 *= ratio;

        shear[0] = -Ft1/kt;

        shear[1] = -Ft2/kt;

```

```

    shear[2] = -Ft3/kt;

}

else{Ft1 = Ft2 = Ft3 = 0.0;}

}

else

{

    if(heating)

    {

        sidata.P_diss                                     +=
gammat*(sidata.vtr1*sidata.vtr1+sidata.vtr2*sidata.vtr2+sidata.vtr3*sidata.vtr3);

        if(heating_track      &&      sidata.is_wall)      cmb-
>tally_pw(gammat*(sidata.vtr1*sidata.vtr1+sidata.vtr2*sidata.vtr2+sidata.vtr3*s
idata.vtr3),sidata.i,sidata.jtype,1);

        if(heating_track      &&      !sidata.is_wall)      cmb-
>tally_pp(gammat*(sidata.vtr1*sidata.vtr1+sidata.vtr2*sidata.vtr2+sidata.vtr3*si
data.vtr3),sidata.i,sidata.j,1);

    }

}

Ft1 -= (gammat*sidata.vtr1);

Ft2 -= (gammat*sidata.vtr2);

Ft3 -= (gammat*sidata.vtr3);

const double Ft_shear_withdamp = sqrt(Ft1 * Ft1 + Ft2 * Ft2 + Ft3 * Ft3);

//double      signofdirection      =
((Ft1*tangentialdirection[0]+Ft2*tangentialdirection[1]+Ft3*tangentialdirection[
2])>0) ? 1 : -1;

const double Ft_coulomb = xmu * fabs(sidata.Fn);

if (Ft_shear_withdamp > losbrechmoment*Ft_coulomb) {

    const double ratiod = Ft_coulomb / Ft_shear_withdamp;

```



```
Ft1 *= ratioid;

Ft2 *= ratioid;

Ft3 *= ratioid;

}

##### End of Edit

// forces & torques

const double tor1 = eny * Ft3 - enz * Ft2;

const double tor2 = enz * Ft1 - enx * Ft3;

const double tor3 = enx * Ft2 - eny * Ft1;

#ifdef SUPERQUADRIC_ACTIVE_FLAG

    double torque_i[3];

    if(sidata.is_non_spherical) {

        double xci[3];

        double Ft_i[3] = { Ft1, Ft2, Ft3 };

        vectorSubtract3D(sidata.contact_point, sidata.pos_i, xci);

        vectorCross3D(xci, Ft_i, torque_i);

    } else {

        torque_i[0] = -sidata.cri * tor1;

        torque_i[1] = -sidata.cri * tor2;

        torque_i[2] = -sidata.cri * tor3;

    }

#endif

// return resulting forces

if(sidata.is_wall) {

    const double area_ratio = sidata.area_ratio;
```

```
i_forces.delta_F[0] += Ft1 * area_ratio;
i_forces.delta_F[1] += Ft2 * area_ratio;
i_forces.delta_F[2] += Ft3 * area_ratio;
#ifdef SUPERQUADRIC_ACTIVE_FLAG
    i_forces.delta_torque[0] += torque_i[0] * area_ratio;
    i_forces.delta_torque[1] += torque_i[1] * area_ratio;
    i_forces.delta_torque[2] += torque_i[2] * area_ratio;
#else
    i_forces.delta_torque[0] = -sidata.cri * tor1 * area_ratio;
    i_forces.delta_torque[1] = -sidata.cri * tor2 * area_ratio;
    i_forces.delta_torque[2] = -sidata.cri * tor3 * area_ratio;
#endif
} else {
    i_forces.delta_F[0] += Ft1;
    i_forces.delta_F[1] += Ft2;
    i_forces.delta_F[2] += Ft3;
    j_forces.delta_F[0] += -Ft1;
    j_forces.delta_F[1] += -Ft2;
    j_forces.delta_F[2] += -Ft3;
#ifdef SUPERQUADRIC_ACTIVE_FLAG
    double torque_j[3];
    if(sidata.is_non_spherical) {
        double xcj[3];
        vectorSubtract3D(sidata.contact_point, sidata.pos_j, xcj);
        double Ft_j[3] = { -Ft1, -Ft2, -Ft3 };
```

```
    vectorCross3D(xcj, Ft_j, torque_j);
} else {
    torque_j[0] = -sidata.crj * tor1;
    torque_j[1] = -sidata.crj * tor2;
    torque_j[2] = -sidata.crj * tor3;
}
i_forces.delta_torque[0] += torque_i[0];
i_forces.delta_torque[1] += torque_i[1];
i_forces.delta_torque[2] += torque_i[2];
j_forces.delta_torque[0] += torque_j[0];
j_forces.delta_torque[1] += torque_j[1];
j_forces.delta_torque[2] += torque_j[2];
#else
    i_forces.delta_torque[0] = -sidata.cri * tor1;
    i_forces.delta_torque[1] = -sidata.cri * tor2;
    i_forces.delta_torque[2] = -sidata.cri * tor3;
    j_forces.delta_torque[0] = -sidata.crj * tor1;
    j_forces.delta_torque[1] = -sidata.crj * tor2;
    j_forces.delta_torque[2] = -sidata.crj * tor3;
#endif
}
}

inline void surfacesClose(SurfacesCloseData & scdata, ForceData&,
ForceData&)
{
    // unset non-touching neighbors
```

```
// TODO even if shearupdate == false?

if(scddata.contact_flags)          *scddata.contact_flags          &=
~CONTACT_TANGENTIAL_MODEL;

double * const shear = &scddata.contact_history[history_offset];

shear[0] = 0.0;
shear[1] = 0.0;
shear[2] = 0.0;
shear[3] = 0.0;
shear[4] = 0.0;
shear[5] = 0.0;
shear[6] = 0.0;
shear[7] = 0.0;
shear[8] = 0.0;
shear[9] = 0.0;
shear[10] = 0.0;
shear[11] = 0.0;
shear[12] = 0.0;
shear[13] = 0.0;
}

inline void beginPass(SurfacesIntersectData&, ForceData&, ForceData&){ }
inline void endPass(SurfacesIntersectData&, ForceData&, ForceData&){ }

protected:

bool heating;

bool heating_track;

class ContactModelBase *cmb;

};
```

```
}  
}  
#endif // TANGENTIAL_MODEL_D AHL_H_  
#endif
```

8.6 MatLab model for pile driving

```
%run.m  
clear all  
close all  
#GUI  
soilmodel='eigen';  
nstrokes=1;  
moledepth=0;           %m   Actual mole depth moledepth per element  
Tipresistance=80;  
Shaftfriction=15;  
global rho_soil  
global phi_soil  
global cohesion  
global grav  
global I_D  
global KO  
%nstrokes=2;           %number of stroke cycles  
ngrid=1;               %Spatial grid resolution  
npoints=12*ngrid+2;   %Number of grid points  
nplot=(npoints-2)/ngrid; %Number of plotted elements  
sim_TIME=500e-3;      %Simulation Time in sec for one stroke cycle
```

```

sim_strokecycle=5;           %Time between two strokes

%time_real=zeros(1,nstrokes);

hitelement=3; %element hit by supportmass

%Length interval:

L_shaft=0.400;           %m Length of mole

deltaL=L_shaft/(npoints-3); %m Space interval

% *****
% *****

%Length of support element and hammer element:

Lsup=2e-3;%unwichtig

Lram=0.5e-3;%unwichtig

%Constants:

grav=9.80665;#3.71;#9.80665;           %m/s^2 Gravitational acceleration

gravity=grav;

%SOIL

E_soil=120e6;           %Pa dynamic Young's modulus of loose sandy soil

nu_soil=0.25;           %1 Poisson ratio of loose sandy soil

G_soil=E_soil/(2*(1+nu_soil)); %Pa shear module of the soil

cohesion=0;

phi_soil=32*(pi/180);

KO=1-sin(phi_soil);

rho_soil=1360;

rho_grain=2720;

%for spherical particles e_max=1 and e_min=0.33

e_max=1;

e_min=0.33;

```

```

e_act=rho_grain/rho_soil-1;
I_D=(e_max-e_act)/(e_max-e_min);
phi_inter_phi_soil=atan(0.3)/phi_soil;%2/3;
%MOLE
E_mole=200e8;          %Pa    Young's modulus of mole material (stainless
steel)
rad_mole=13.5e-3;      %m     Radius of mole
A_mole=pi*rad_mole^2; %m     Cross section area of mole
alpha=60/180*pi;      %rad   Sohlneigung, tip angle
effectiveDepth=2;%10*rad_mole; %loose soil <10*D dense soil <20*D
%Geometry:
%Note: positive x-direction is from the surface into the soil!
A_cushion=pi*rad_mole^2; %m    Cross section area of cushion block
L_cushion=0.0138;%deltaL; %m    Thickness of cushion block (same as other
elements)
%L_cushion=deltaL;    %m     Thickness of cushion block (same as other
elements)
%Global parameters:
%Spring properties
Kf=6222;              %N/m   Spring constant of percussion spring
Kb=73;                %N/m   Spring constant of support spring
Lfu=35.00e-3;        %m     Uncompressed length of percussion spring
Lfp=20.00e-3;        %m     Precompressed length of percussion spring
Lfc=20.00e-3;        %m     Compressed length of percussion spring
Lbu=108.00e-3;       %m     Uncompressed length of support spring
Lbp=52.35e-3;        %m     Precompressed length of support spring
Lbc=29.35e-3;        %m     Compressed length of support spring

```

%Masses:

M_casing=276e-3; %kg Total mass of housing

M_shaftsegment=M_casing/(npoints-3); %kg Mass of one any interior tube segment;

M_tip=24e-3;%2*M_shaftsegment; %kg Mass of MOLE tip;

M_ram=110e-3; %kg Mass of ram (hammer)

M_support=460e-3; %kg Support mass

%Coefficients of restitution for hammer and tube cap:

e1=0.1;

e2=0.1;

%Calculate friction resistance along tube segments:

 %kg/m³ Soil density

%phi_friction=atan(0.4);%14*(pi/180); %rad Angle of friction
between soil and PEN-tube

%tanphi=tan(phi_friction);

sigma=zeros(1,npoints);

tip=npoints-2;

hammer=npoints-1;

support=npoints;

%Spring constants:

 E_cushion=E_mole; %Pa Young's modulus of PEN-tube top element;

 J_tip=0.5; % s/m Soil damping constant at the tip

 J_wall=0.15; % s/m Soil damping constant along the
moles's side wall (friction)

K_shaftsegment=(A_mole*E_mole)/deltaL; %N/m Spring constant of PEN-
tube segments

$K_{\text{cushion}}=(A_{\text{cushion}}*E_{\text{cushion}})/L_{\text{cushion}};$ %N/m Spring constant of cushion material

$\text{check_Irho}=(e_{\text{act}}<e_{\text{min}})+(e_{\text{act}}>e_{\text{max}});$

if $\text{check_Irho}>0$

error('Your density index is too large or too small, check your rho_soil and rho_grain')

end

$\text{deltat}=2*\text{pi}*\text{sqrt}(M_{\text{shaftsegment}}/(K_{\text{shaftsegment}}))/40;$ %s Timestep

$\text{ntmax}=\text{round}(\text{sim_TIME}/\text{deltat});$ %Number of timesteps

$\text{time}=(0:\text{deltat}:\text{deltat}*(\text{ntmax}));$ %s Time

%-----Boundary conditions-----

%Initial conditions:

$\text{mass}=\text{zeros}(1,\text{npoints});$ %mass in kg

$\text{mass}(\text{npoints}-1)=M_{\text{ram}};$

$\text{mass}(1:\text{npoints}-3)=M_{\text{shaftsegment}};$

$\text{mass}(\text{npoints}-2)=M_{\text{tip}};$

$\text{mass}(\text{npoints})=M_{\text{support}};$

%

$K=\text{zeros}(1,\text{npoints});$ % stiffness of pile

$K(1:\text{npoints}-3)=K_{\text{shaftsegment}};$

$K(\text{npoints}-1)=K_f;$

$K(\text{npoints})=K_b;$

$K(\text{npoints}-2)=K_{\text{cushion}};$

$K_{\text{soil}}=\text{zeros}(1,\text{npoints}-2);$ %stiffness of soil

$J_{\text{soil}}=\text{zeros}(1,\text{npoints}-2);$ %soil damping

$J_{\text{soil}}(1:\text{npoints}-3)=J_{\text{wall}};$

```
Jsoil(npoints-2)=J_tip;
V=zeros(ntmax,npoints);
D=zeros(ntmax,npoints);
    D(1,npoints-1)=-(Lfu-Lfp);
C=zeros(1,npoints);
F=zeros(ntmax,npoints);
R=zeros(ntmax,npoints);
Z=zeros(ntmax,npoints);
Dsoil=zeros(ntmax,npoints);
hitforce=zeros(ntmax,1);
hitoverlap=0;
% moledepth_init=0.5;           % m    Initial depth of mole upper end in
soil
% initialpositions=[D(1,:)';zeros(npoints,1)];
% for ii=2:npoints-2
% initialpositions(ii,1)=initialpositions(ii-1,1)+deltaL;
% end
% initialpositions(npoints-1,1)= D(1,npoints-1)+L_shaft;
% initialpositions(npoints,1)=D(1,npoints)+Lbu;
% initialpositions(npoints,2)=1;
% initialpositions(npoints-1,2)=1;
% %plot initial conditions
% scatter(initialpositions(:,2),initialpositions(:,1))
% xlim([-4, 5])
% set(gca,'YDir','reverse');
flag=0;
```

```
flags=zeros(npoints-3,1);
C1max=C(npoints-2);
C2max=hitoverlap;
tipDepth=zeros(nstrokes,1);
% soil layers
layer.number_of_layers=2;
layer.top=zeros(layer.number_of_layers,1);
layer.bot=zeros(layer.number_of_layers,1);
layer.rho=zeros(layer.number_of_layers,1);
layer.phi_soil=zeros(layer.number_of_layers,1);
layer.cohesion=zeros(layer.number_of_layers,1);
% cohesion dry = [0 - 5000] cohesion wet = [0 - 15000]
% cohesion wet clay = [10000 - 40000]
% _____ user input
layer.top(1)=11;
layer.bot(1)=12;
layer.rho(1)=1460;
layer.phi_soil(1)=phi_soil;
layer.cohesion(1)=cohesion;
layer.e_max(1)=1;
layer.e_min(1)=0.33;
layer.rho_grain(1)=2720;
layer.top(2)=15;
layer.bot(2)=20;
layer.rho(2)=1600;
```

```

layer.phi_soil(2)=phi_soil;

layer.cohesion(2)=cohesion;

layer.e_max(2)=1;

layer.e_min(2)=0.33;

layer.rho_grain(2)=2720;

%-----

for il=1:layer.number_of_layers

layer.I_D(il)=(layer.e_max(il)-(layer.rho_grain(il)/layer.rho(il)-
1))/(layer.e_max(il)-layer.e_min(il));

end

check_rho=find((layer.I_D<0)+(layer.I_D>1));

if check_rho>0

    error(['Your density index is too large or too small, check your rho_soil and
rho_grain of layer ' num2str(check_rho)])

end

if length(layer.top)~=layer.number_of_layers

    error('The variable layer.number_of_layers does not correspond to the number
of defined layers')

end

for stroke=1:nstrokes

%[rho_soil_u,phi_soil_u,cohesion_u]=get_soilprop_u(moledepth+deltaL+(npoints-3-1)*deltaL,layer); %rho_soil under tip

for i=1:npoints-3

%
[rho_soil_i,phi_soil_i,cohesion_i,KO_i]=get_soilprop_i(moledepth+deltaL/2+(i-1)*deltaL,layer);

%   sigma(i)=sigma_v(moledepth+deltaL/2+(i-1)*deltaL, layer);

    sigma(i)=(moledepth+deltaL/2+(i-1)*deltaL)*rho_soil*grav; %ersatz

```

```

% phi_inter=phi_soil_i*phi_inter_phi_soil;

Ru_shaftsegment(i)=Shaftfriction/(npoints-
3);%KO_i*sigma(i)*rad_mole*(2*pi)*deltaL*tan(phi_inter); %K_0 consider
end;

tipposition=moledepth+deltaL+(npoints-3-1)*deltaL;

% sigma(npoints-2)=sigma_v(tipposition, layer); % flat tip

%trailing cable


%Ru_shaftsegment(1)=Ru_shaftsegment(1)+moledepth*2*pi*rad_mole*sigma_
v(moledepth, layer)/2*0.1;

%Mohr-Coulomb criteria

yield_soil=sigma(npoints-3)*(1+KO)/2*sin(phi_soil)+cohesion*cos(phi_soil);
%Pa yield stress of the soil

shear_soilpile=yield_soil/10; %Pa shear stress at soil-pile interface

alpha=60/180*pi;

switch soilmodel
case 'Grundbruch nach NORM'
% bearingcapacityflattip %for layered soil just bearingcapacityflattip
%Qu_soil=3/4*Qu_soil;

case 'lokales Scherversagen nach MC'

% mohrcoulomb3

case 'Meyerhof'

% bearingcapacityMeyerhof

case 'eigen'

Qu_soil=Tipresistance/A_mole;

end

Ru_tip=Qu_soil*A_mole; %N Ultimate soil resistance at mole tip

```

```

Q_tip=(1+nu_soil)/(2*E_soil)*yield_soil*rad_mole;           %m Soil quake
at tip calculated from material parameters (maximal elastic ground deformation)

rad_disturbed=rad_mole*10;                                %m Disturbed region
around mole ;

Q_shaft=shear_soilpile/G_soil*rad_mole*log(rad_disturbed/rad_mole); %m
Soil quake at boundary segments calculated from material parameters

K_tip=Ru_tip/Q_tip;           %N/m   Spring constant of soil at the tip

%end of updating soil properties

%

Ksoil(npoints-2)=K_tip;

K_wall=zeros(1,npoints);

for i=1:npoints-3

    K_wall(i)=Ru_shaftsegment(i)/Q_shaft;   %N/m   Spring constants of soil
along shaft segments

end;

Ksoil(1:npoints-3)=K_wall(1:npoints-3);

for n=1:ntmax           % do timestep

    %-----Schlag Top Druck ist positiv

    C(npoints-1)=(D(n,npoints) - D(n,npoints-1)); %percussion spring

    F(n,npoints-1) = C(npoints-1)*K(npoints-1);

    C(npoints)=(D(n,1) - D(n,npoints)+(Lbu-Lbp)); %support spring

    F(n,npoints) = C(npoints)*K(npoints);

    %Federn koennen nicht kuerzer als compressed length sein:

        if C(npoints)>(Lbu-Lbc)

            D(n,npoints)=D(n,1)-(Lbp-Lbc);

            C(npoints)=(Lbu-Lbc);

            F(n,npoints) = C(npoints)*K(npoints);

```

```

        end

        if C(npoints-1)>(Lfu-Lfc)

            D(n,npoints-1)=D(n,npoints)-(Lfu-Lfc);

            C(npoints-1)=(Lfu-Lfc);

            F(n,npoints-1) = C(npoints-1)*K(npoints-1);

        end

    for i=1:npoints-3    % calculate all mole springforces in this timestep

        C(i) = D(n,i) - D(n,i+1);

        F(n,i) = C(i)*K(i);

    end;

    C(npoints-2)=(D(n,npoints-1) - D(n,npoints-2));           %cushion spring
    (hammer-tip contact)

    if C(npoints-2)>=0

        m_mean1=mass(npoints-1)*mass(npoints-2)/(mass(npoints-
        2)+mass(npoints-1));

        gamma_n1=sqrt(4*m_mean1*K_cushion/(1+(pi/log(e1))^2));

        F(n,npoints-2)=K(npoints-2)*C(npoints-2) - gamma_n1*(V(n,npoints-2)-
        V(n,npoints-1));    %step (a) and step (d)

        if F(n,npoints-2)<0

            F(n,npoints-2)=0;    %step (f) no tension of spring

        end

    else

        F(n,npoints-2)=0;

    end

    hitoverlap=D(n,npoints)-D(n,hitelement);

    if(hitoverlap<=0)

        hitforce(n)=0;

```

```

else
m_mean=mass(hitelement)*mass(npoints)/(mass(hitelement)+mass(npoints));
    gamma_n=sqrt(4*m_mean*K_cushion/(1+(pi/log(e2))^2));
    hitforce(n)=hitoverlap*K_cushion - gamma_n*(V(n,hitelement)-
V(n,npoints)); %step (a) and step (d)
    if hitforce(n)<0
        hitforce(n)=0; %step (f) no tension of spring
    end
end
    %Soil resistance at tip
if (Dsoil(n,npoints-2)<(D(n,npoints-2)-Q_tip))
    Dsoil(n,npoints-2)=D(n,npoints-2)-Q_tip;
    flag=1;
end
    if flag==0
        R(n,npoints-2)=(D(n,npoints-2)-Dsoil(n,npoints-2))*Ksoil(npoints-2)* (1 +
Jsoil(npoints-2)*V(n,npoints-2));
    elseif flag==1
        R(n,npoints-2)=(D(n,npoints-2)-Dsoil(n,npoints-2))*Ksoil(npoints-2) +
Jsoil(npoints-2)* Ksoil(npoints-2)* Q_tip*V(n,npoints-2);
    end
    if R(n,npoints-2)<0
        R(n,npoints-2)=0;
    end
    Dsoil(n+1,npoints-2)=Dsoil(n,npoints-2);
    %Soil resistance and plastic displacement via friction along tube
    for i=1:npoints-3

```



```

if(Dsoil(n,i)>(D(n,i)+Q_shaft)) %Routine #1 plastic displacement

Dsoil(n,i)=D(n,i)+Q_shaft;

flags(i)=1;

elseif(Dsoil(n,i)<(D(n,i)-Q_shaft))

Dsoil(n,i)=D(n,i)-Q_shaft;

flags(i)=1;

end

if flags(i)==0 %linear elastic

R(n,i)=(D(n,i)-Dsoil(n,i))*Ksoil(i)* (1+Jsoil(i)*V(n,i));

elseif flags(i)==1

R(n,i)=(D(n,i)-Dsoil(n,i))*Ksoil(i) + Jsoil(i)* Ksoil(i)* Q_shaft*V(n,i);

end

Dsoil(n+1,i)=Dsoil(n,i);

end

% ::::::::::::::::::::::::::::::::::::::INTEGRATION::::::::::::::::::::::::::::

Z(n,1) = -F(n,npoints)-F(n,1)-R(n,1) + gravity*mass(1); %Druckkraft ist
positiv!!

for i=2:npoints-3

if(i==hitelement)

Z(n,i)=F(n,i-1)-F(n,i)-R(n,i) + hitforce(n) + gravity*mass(i);

else

Z(n,i)=F(n,i-1)-F(n,i)-R(n,i) + gravity*mass(i);

end

end

Z(n,npoints-2)=F(n,npoints-3)+F(n,npoints-2)-R(n,npoints-2) +
gravity*mass(npoints-2); %tip

```

```

    Z(n,npoints-1)=-F(n,npoints-2)+F(n,npoints-1) + gravity*mass(npoints-1);
%hammer

    Z(n,npoints)=-F(n,npoints-1)+F(n,npoints) - hitforce(n) +
gravity*mass(npoints); %support mass

    %

    % Velocities:

    V(n+1,:) = V(n,:) + Z(n,:)./mass*deltat;

    D(n+1,:) = D(n,:) + deltat*V(n+1,:);

    PotEnergy=1/2*Kf*C(npoints-1)*C(npoints-
1)+1/2*Kb*C(npoints)*C(npoints);

    KinEnergy=1/2*M_ram*V(n,npoints-1)*V(n,npoints-
1)+1/2*M_support*V(n,npoints)*V(n,npoints);

    TotEnergy=PotEnergy+KinEnergy;

end

disp('finished stroke')

disp(stroke)

moledepth=moledepth+D(end,npoints-2);

tipDepth(stroke)=moledepth+L_shaft;

end

% plot(tipDepth,'x')

realtime=0:3.75:3.75*nstrokes-1;

savefile=strcat('output.mat');

save(savefile, 'D', 'time')

totset=D(end,12);

#save('-ascii','-append', 'results/results.txt','counter', 'totset')

%plots

plot(time,D(:,12),'linewidth',2) %tip displacements

```

```
hold on

plot(time,D(:,13),'r','linewidth',2) %hammer displacements
plot(time,D(:,14),'g','linewidth',2) %supportmass displacements
plot(time,Dsoil(:,npoints-2),'k','linewidth',2)
set(gca,'YDir','reverse');
h=legend('Tip','Soil')#,'hammer','support')
xlabel('Time [s]','fontsize',16)
ylabel('Displacements [m]','fontsize',16)
grid on
set (h, "fontsize", 14)
set(gca, "fontsize", 12)
xlim([0,0.14])
ylim([-0.005,0.006])
#ylim([-0.016,0.006])
% plot(F(:,12)) %force on tip
% plot(Dsoil(:,1),'k')
% hold on
% plot(D(:,1))
plot(f04reibung.time,f04reibung.D(:,12),'linewidth',2)
hold on
set(gca,'YDir','reverse');
plot(f03reibung.time,f03reibung.D(:,12),'r','linewidth',2)
plot(f02reibung.time,f02reibung.D(:,12),'g','linewidth',2)
plot(f01reibung.time,f01reibung.D(:,12),'k','linewidth',2)
%plot(f00reibung.time,f00reibung.D(:,12),'c','linewidth',2)
```

```
legend('\mu_{inter}=0.4','\mu_{inter}=0.3','\mu_{inter}=0.2','\mu_{inter}=0.1')
```

```
xlabel('Time [s]')
```

```
ylabel('Displacements [m]')
```