

TWO-PHASE DYNAMICS OF
GRANULAR PARTICLES IN A
NEWTONIAN FLUID

SHI HAN NG

THE UNIVERSITY OF
ELECTRO-COMMUNICATIONS
GRADUATE SCHOOL OF INFORMATICS
AND ENGINEERING

A DISSERTATION SUBMITTED FOR
DOCTOR OF PHILOSOPHY IN
ENGINEERING

JUNE 2015

TWO-PHASE DYNAMICS OF GRANULAR PARTICLES IN A NEWTONIAN FLUID

APPROVED BY SUPERVISORY COMMITTEE:

CHAIRPERSON: ASSOC. PROF. Hans-Georg Matuttis

MEMBER: PROF. Takeshi Miyazaki

MEMBER: PROF. Hiroshi Maekawa

MEMBER: PROF. Tomio Okawa

MEMBER: PROF. Naruo Sasaki

Copyright
by
SHI HAN NG
2015

ニュートン流体における粉体の二相動力学

ウ　ン　シ　ハ　ン
Ng Shi Han

概 要

自然界では、粉体・流体における固液混相流現象は砂漣のような無害の現象から雪崩、地滑り、地震による液状化などの自然災害に及ぶ。このような現象の解明が重要であるにも関わらず、任意の形状を持った粒子や流体領域を考慮に入れるシミュレーションはほとんどない。従って、流体内における粒子の運動を解析するには離散要素法を固液混相流シミュレーションに導入するのは困難である。このような固液二相流の問題を解析するために、巨視的な及びメゾスコピックな物理量が得られるシミュレーション方法が必要とされる。本研究では、流体シミュレーションの有限要素法を粉体シミュレーションの離散要素法と組み合わせることによって、非圧縮性ニュートン流体中における粉体の微視的シミュレーションを開発した。「微視的シミュレーション」とは本研究で開発されたモデルは流体が「粒子を通して流れる」ではなくて「粒子の周りを沿って流れる」ものを意味する。

粉体シミュレーションでは、離散要素法 (DEM) では接触している多角形粒子が変形しないと仮定し、その弾性接触力は重なり面積に比例しているとされている。粉体の運動方程式の解を Gear の予測子・修正子法 (二次後退差分法 BDF2) を用いて求める。一方、流体シミュレーションでは、流体の運動を示す非圧縮ナビエーストークス方程式をガラキン有限要素法 (FEM) で微分代数方程式 (DAE) に定式化する。その時間方向には粉体相と同じく二次後退差分法を用いて離散化する。得られた非線形連立 1 次代数方程式を Newton-Raphson 法で解く。また、空間方向の離散化では、Delaunay 三角形分割かつ緩和計算で後処理した三角要素を Taylor-Hood 要素として使用する。適切な境界条件および粉体の境界を沿って流体の応力テンソルを積分して得た抗力を利用し、粉体の DEM と流体の FEM を結合する。シミュレーションの検証としては、流体中に落下する粒子の壁からの補正係数を計算する。

流体のシミュレーションを自由表面のシミュレーションに拡張する。有限要素法から得られた流体境界上の流速を Adams-Bashforth 解法で積分し、流体境界を移動させることによって、自由表面の運動を求める。従来型のシミュレーション方法に比較するとこの方法は必要なデータ構造などの付加的な取り組みを最小限にすることができる。検証として、水柱の崩壊のシミュレーションを行い、理論および実験データと比較する。

DEM-FEM シミュレーションを用いて二つの数値的実験を行った：1) 圧密に関するシミュレーションから、粒状集合に流体を加えることにより、その系の音速を流体に含まれていない系より増加させることが示された。また、流体を加えることによって、圧密が減速させることが確認できた。2) 粒子の柱のシミュレーションにより、水中において粒子の回転に関する重要さが失われることが示された。

Two-phase dynamics of granular particles in a Newtonian fluid

Shi Han Ng

ABSTRACT

Many scientific and technical problems which concern the dynamics of complex fluids such as multi-phase-flow and realistic flow in porous and granular media deal with the interaction between fluids and particles, rather than with the dynamics of the fluid alone. The research of how the surrounding fluid affects the dynamics of particles, or how to deal with the problem computationally for the microscopic level is still at the beginning. The aim of this study is to develop a microscopic simulation method (fluid goes around the particles) where granular particles can be simulated inside fluids to study those problems. This is done by combining the simulation method for granular particles with the simulation method for the incompressible Newtonian fluid.

The granular particles are implemented via the discrete element method (DEM) where the elastic contact force between two undeformed contacting polygonal particles is proportional to the overlap area (“hard particle, soft contact”). The Gear Predictor–Corrector of 2nd-order (BDF2) is used as the time integrator to solve the equations of motion of the particles. For the fluid phase, the implementation of the incompressible Navier–Stokes equations via the Galerkin finite element method (FEM) is formulated as differential algebraic equations (DAE) with the pressures as the Lagrange parameters. The time integration is again via the BDF2 while the resulting non-linear equations are solved via the Newton–Raphson methods. The spatial discretization is via the Taylor–Hood elements from Delaunay triangulations with additional post-processing with the relaxation algorithm. The coupling of the DEM for the granular particles and the FEM for the fluid is via appropriate boundary conditions and the drag force (computed by the integration of the fluid stress tensor over the particle’s surface). This is being verified via the computation of wall correction factors of a sinking particle.

The fluid simulation is extended to a simulation of free surfaces where the motion of the surface is integrated out according to the velocity on the surface which is obtained from the FEM-scheme. The second-order Adams–Bashforth method turns out to be the most suitable integrator for the surface motion. Compared to

conventional efforts, which try to solve partial differential equations for the motion of the surface, the additional effort in our method with respect to new data structures etc. is minimal. The free surfaces code is verified by simulating the collapse of a water column. For the speed of the wavefronts, excellent agreement is obtained for large viscosity with the lubrication approximation. The agreement of the results with the experimental data for water is a further gratifying result.

Two numerical experiments are conducted using the DEM-FEM code: one with a rather slow dynamics, another one relatively more “violent”. The compaction simulation has shown that the addition of fluid to a granular assembly can increase the sound velocity in the system, compared to the dry case. The high viscosity slowed down the compaction, irrespective whether the system was tapped only on the ground or on the whole boundary. The granular column simulations show that for systems immersed under fluids, rolling of particles becomes less important than for the corresponding dry systems.

Contents

Contents	i
List of Figures	v
List of Tables	xi
1 Introduction	1
1.1 Intended Characteristics of the Simulation	1
1.2 Microscopic and Macroscopic Simulation	3
1.3 Simulation of the Particle Part	4
1.4 Simulation of the Fluid Part	4
1.4.1 Particle Methods for Fluid	5
1.4.2 Grid Methods	6
1.4.3 Finite Element Methods	9
1.5 Performance Issues	10
1.6 Overview of the Thesis	11
2 Simulation of the Granular Part	12
2.1 The Principle of Discrete Element Method	12
2.1.1 Elastic Force in Normal Direction	14
2.1.2 Dissipative/Damping Force in Normal Direction	16
2.1.3 Friction in Tangential Direction	18
2.1.4 Total Force and Torque	21
2.2 Polygonal Particles	22
2.2.1 Creating the Polygonal Particles	23
2.2.2 Mass, Center of Mass and Moment of Inertia	24
2.2.3 Overlap Area	27
2.2.4 Force Directions and Force Points	30

2.2.5	Dealing with Penetrating Contacts	31
2.3	Time Integrator	33
3	Simulation of the Fluid Part	38
3.1	Governing Equations	38
3.2	Finite Element Methods	39
3.2.1	The Weak Form of the Navier–Stokes Equations	40
3.2.2	Discretization of the Weak Form	43
3.2.3	Choice of Elements	46
3.2.4	Taylor–Hood Element	46
3.2.5	Time Integrator	53
3.2.6	Newton–Raphson Method	55
3.3	Automatic Mesh Generation	62
3.3.1	Constrained Delaunay Triangulation	62
3.3.2	Relaxation Algorithm	64
4	Simulation of Flow Problems with Free Boundaries	69
4.1	Philosophy of the Surface Computation	69
4.2	Surface Modeling via ODE-Integrators	71
4.2.1	Choosing the Integrators	71
4.2.2	Using Interpolated Velocities	73
4.3	Boundary Conditions	75
4.4	Simulation of the Collapse of a Water Column	77
4.4.1	Water Column with High Viscosity	77
4.4.2	Water Column with Low Viscosity	79
5	Simulation of Fluid with Suspended Particles	83
5.1	Coupling the Discrete Element Method and the Finite Element Method	83
5.1.1	Form Drag	86
5.1.2	Friction Drag	88
5.1.3	Modeling a Shadow around the Particle	89
5.2	Stability and consistence: Two Square Particles in Fluid	91
5.2.1	Verification with Wall Correction Factor	94

6	Applications of the Code	98
6.1	Effect of the Surrounding fluid on the Compaction of Granular Materials by Tapping	98
6.1.1	Previous Studies	99
6.1.2	Outline of this Study	100
6.1.3	Initialization and preparation of the System	100
6.1.4	Tapping	102
6.1.5	Evolution of the center of mass	104
6.1.6	Results	105
6.1.7	Summary	108
6.2	Collapse of Granular Column in Fluid	108
6.3	Sedimentation of Multiple Particles in a Fluid	113
7	Epilogue: Beyond the Scope of this Thesis	114
7.1	Performance improvements for future simulations	114
7.2	Problems which can be tackled by the program in its current form . .	115
7.3	Desirable extensions for future simulations	117
7.3.1	Extensions which would improve speed, accuracy or stability .	117
7.3.2	Extensions which make additional phenomenology accessible for simulation	119
7.4	Possible extensions	121
7.5	Future simulation topics	122
8	Summary	124
A	Weak Form of the Navier–Stokes Equations	126
B	Differential-Algebraic Equations	130
B.1	Polar Coordinate System	130
B.2	Cartesian Coordinate System with DAE	132
C	Using MEX-Files in MATLAB	135
C.1	Outer Product Computation of two Vectors	135
C.2	Writing the Fortran/MEX-file	137
C.2.1	Preparation	137
C.2.2	Declaring the Variables	137
C.2.3	Reading Input from MATLAB	139

C.2.4	Actual Computation	140
C.2.5	Output to MATLAB	140
C.2.6	Actual Computation Subroutine	141
C.3	Compiling and Using Fortran/MEX-file	141
C.4	Remark	142
References		145
List of Publications Related to the Thesis		153
Author Biography		155
Acknowledgment		157

List of Figures

1.1	Intended region of validity for our simulation of granular particles inside Newtonian fluid. The method should be applicable for technical problems as well as well as for systems in the geosciences.	2
1.2	Microscopic modeling where the flow goes around the particles and macroscopic modeling where the flow goes through the particles. . . .	3
1.3	Resolving the boundary of a solid by the specification of interpolated mirror condition inside the solid.	7
1.4	Discretization of a circle via square grid: only the approximation of the area is improved with finer grid but not the circumference. . . .	7
1.5	Relative position and orientation on finite difference square grid. . . .	7
2.1	Normal and tangential directions of two contacting round particles and location of the force point.	13
2.2	Relationships between the elastic force, contact area, and penetration depth for three types of contact based on the elasticity theory. . . .	14
2.3	Overlapping particle-pair with overlap area.	15
2.4	(a) Direct summation of the elastic $F_{c,\perp}$ and damping force $F_{d,\perp}$ in normal direction leads to jumps in total force F_{\perp} during the approach and separation of the contacting particles. (b) Correction in the damping force is performed to cut off the unphysical attractive force during separation.	17
2.5	Two kinds of friction in sliding contact of particles: static friction as constraint force and dynamic friction as dissipative force.	19
2.6	Obtaining the relative tangential velocity $v_{\text{rel},\parallel}$ from velocities $\mathbf{v}_1, \mathbf{v}_2$, angular velocities ω_1, ω_2 of the contacting particles and the vectors $\mathbf{r}_1, \mathbf{r}_2$ between their centroids and the force point P	20
2.7	Implementation of Cundall–Strack model for the tangential friction force in the discrete element simulation.	20
2.8	Decomposition of the normal forces F_{\perp} (elastic $F_{c,\perp}$ and damping $F_{d,\perp}$ forces) and tangential friction force $F_{d,\parallel}$ between two contacting particles into horizontal and vertical components.	22

2.9	(a) Constructing a polygon by inscribing a regular polygon with eight vertices into an ellipse. (b) Constructing a polygon via convex hull for a set of random points.	24
2.10	(a) Decomposition of a particle with $n = 10$ vertices into n triangles. (b) Triangle enlarged from (a) to show the computation of the moment of inertia using the radial vector \mathbf{r}_i and edge vector \mathbf{s}_i	25
2.11	“Bounding boxes” constructed from the endpoints of the edges. . . .	28
2.12	Construction of four triangles by connecting each edge to the endpoints of the other edge.	29
2.13	Computation of the intersection points of the intersecting edges. . . .	30
2.14	Overlap geometry with area of two contacting polygonal particles constructed via the intersection points and the vertices of the particles which penetrate into the other particle.	30
2.15	(a) The normal \hat{n} and tangential \hat{t} directions of two contacting polygonal particles obtained directly from the “contact line” $\overline{S_1S_2}$. (b) Normal direction \hat{n} as the weighted average of \hat{n}_1 and \hat{n}_2 . The force point P is located at the center of mass of the overlap geometry. . . .	31
2.16	Exaggerated sketches showing the resulting four intersection points from a penetrating contact.	31
2.17	Tangential \hat{t} and normal direction \hat{n} at force point P computed from the cut-off vectors \mathbf{d}_1 and \mathbf{d}_2 of a penetrating contact.	32
3.1	Triangular element with vertices in counterclockwise orientation. . . .	48
3.2	A triangle with area Δ_e partitioned into three smaller triangles, where Δ_1 is the area of $\triangle PP_2P_3$, Δ_2 is the area of $\triangle PP_3P_1$, and Δ_3 is the area of $\triangle PP_1P_2$	49
3.3	Vertices and midpoints of a triangle in local area coordinates.	50
3.4	First-order piecewise polynomial (affine) shape functions $\psi_1(x, y)$, $\psi_2(x, y)$ and $\psi_3(x, y)$	51
3.5	Second-order piecewise polynomial (quadratic) shape functions. . . .	52
3.6	Geometrical interpretation of the Newton–Raphson method.	56
3.7	The Jacobian matrix for the Newton–Raphson method when solving flow problem with FEM.	60
3.8	Profiling results of solving the linear system $Ax = b$ with matrices A of different number of nonzero elements using BiCGSTAB(l), BiCGSTAB(l) with Cuthill–McKee and UMFPACK.	61
3.9	Delaunay triangulation of set of a randomly positioned grid points inside a rectangular domain.	63
3.10	Mesh generation around particles with (a) mathematically exact Delaunay triangulation, where triangles are cutting through a particle and (b) constrained Delaunay triangulation.	64

3.11	Convergence of randomly spaced n points on the y -axis with interactions of linear springs between neighboring points towards the equilibrium (equidistant spacing) using the integrator from Eq. (3.97).	66
3.12	Edges of triangular mesh treated as linear spring for relaxation algorithm.	66
3.13	(a) Constrained Delaunay triangulation and (b) the application of the relaxation algorithm.	68
3.14	Grid adaption by applying the relaxation algorithm on a triangular mesh of a square grid with a Gaussian potential.	68
4.1	Free surface modeling: moving the surface of the fluid using the obtained velocities from the finite element method.	71
4.2	Snapshots from the time evolution of the gravity wave with the triangular meshes in (a), pressure distribution in (b), flow velocities in (c) and (d).	73
4.3	Time evolution of the volume for the surface integration with Euler and Adams–Bashforth of second order.	74
4.4	Choosing the velocity data for the ODE-integrator to compute the surface node for the next time-step.	75
4.5	Treatment of boundary conditions near the between the surface and the solid wall.	75
4.6	Comparison between flow at at no-slip boundary and the rising of the surface near a no-slip boundary.	76
4.7	(a) Hydrostatic pressure profile which inhibits the movement of the water column resulting from setting the boundary conditions of the pressures. (b) Every node on the surface is set to zero which is the “correct” pressure profile in the moment where the wall is removed and the water step is allowed to flow towards the right.	78
4.8	Time evolution for the fronts in for the collapse of the square water column at $\nu = 10^{-1}$ [m ² /s].	78
4.9	Comparison of the time evolution of the advancing wave front under the lubrication approximation with our simulation $\nu = 10^{-1}$ [m ² /s].	79
4.10	Advance of the fluid front according to the experiment of Martin and Moyce [1] for the collapse of a fluid column of 57×57 [mm] at $\nu = 10^{-6}$ [m ² /s].	79
4.11	Time evolution for the fronts in for the collapse of the square water column at $\nu = 10^{-6}$ [m ² /s] and step-size $\Delta t = 1.0 \times 10^{-5}$	80
4.12	Comparison of flow patterns at the upper right corner of the column for high and low viscosity at $t = 0.0375$ [s].	81
4.13	Snapshots showing the shape of the wave-front for low viscosity flow.	81

4.14	Comparison of the computed time evolution of the advance of the fronts with various viscosities with the experimental data of Martin and Moyce [1].	82
5.1	Non-slip boundary conditions on the nodes around the particles. . . .	83
5.2	Flowchart of the simulation fluid with suspended polygonal particles. . .	84
5.3	The effect of the form drag and friction drag on the object immersed in fluid depends on the shape and angle of attack.	86
5.4	Computation of the fluid force on the particle from the nodal values (u_i, v_i, p_i) of an FEM-element Δ_e	87
5.5	Separated space in straightforward modeling of two-dimensional particles without shadow in (a), (b) Fluid space (white), shadow by which the particles interact (light gray) and core (dark gray) which forms the boundary of the fluid flow and (c) Three-dimensional arrangement of grains which leads to flow between the particles along the thick lines which is supposed to be mimicked by the shadow on the left.	90
5.6	Pore space formed by particle asperities of the rods which are equivalent to the two-dimensional particles.	90
5.7	Snapshots with velocity fields and pressure distributions of two square particles sinking under the influence of gravity after being released in a symmetrical configuration.	92
5.8	Above: Form drag and friction drag from the fluid on the left particle. Below: Center of mass (in y direction) of the left particle for both simulation with and without fluid.	93
5.9	Dimensions of the system for computing the wall correction factor. . .	94
5.10	Initial orientations of the polygon for computing the wall correction factor with corner-down and edge-down.	94
5.11	Left: Vertical velocities for a sinking particle via our DEM-FEM code for diameters $D = 3$ [cm], $\mu = 1.0$ [Pa · s], $Re \approx 35$. Right: Terminal velocity after Ristow [2, Fig. 2] via a MAC-scheme.	96
5.12	Snapshots with velocity fields and pressure distributions of sinking a dodecagon under the influence of gravity for the computation of the terminal velocity.	97
6.1	(a) Initialization of the system with 195 particles. (b) system with the particles settled down which is used as initial configuration. (c) Enlarged region of (b) showing the pore space and triangular meshes around the particles.	101
6.2	Time evolution of the acceleration of a particle in the lowest layer of the dry simulation and wet simulation as well as intensity of the original pulse.	103

6.3	Position of the center of mass for reducing the coefficient of friction μ .	104
6.4	“Final” potential energy of the wide system with different coefficients of friction μ in double logarithmic plots and the fitting.	105
6.5	Shock propagation through the system for the dry and the immersed system for tapping of the bottom.	106
6.6	The incompressible fluid helps to stiffen the interparticle contacts and leads to higher sound velocity.	106
6.7	Compaction of system with tapping only the bottom for the dry and the immersed particles.	107
6.8	Compaction of system for with tapping the whole wall for the dry and the immersed particles.	108
6.9	Preparation of the granular columns with loose packing $\phi = 0.767$, as well as dense packing $\phi = 0.793$	109
6.10	The distribution of the number of contacts \bar{n}_c for the granular columns with different volume fractions ϕ and number of particles n	110
6.11	Snapshots of the collapse of granular column in fluid for system with volume fraction of 0.789.	110
6.12	Time evolution of the front position for the dry system and the system with particles immersed in fluid with different volume fraction.	111
6.13	Sedimentation of 252 polygonal particles of different shapes in a fluid domain of size 135.6×90 [mm] with about 3700 triangular elements. .	112
B.1	Simple pendulum problem with point mass m suspended from a string with length l from the origin.	131
C.1	Time consumption of both MATLAB and Fortran/MEX version outer product functions for different vector sizes for MATLAB version R2014a on Mac OS X 10.10.	143

List of Tables

2.1	Gear corrector coefficients for second-order differential equations of the form $\ddot{\mathbf{r}} = f(\mathbf{r}, \dot{\mathbf{r}})$	36
3.1	List of triangular elements which are LBB stable.	47
5.1	Comparison between our numerical and Richou et al.'s results of wall correction factor $\lambda(k)$ for $k = 0.1250$	96

Chapter 1

Introduction

Many scientific and technical problems which concern the dynamics of complex fluids such as multi-phase-flow and realistic flow in porous and granular media deal with the interaction between fluids and particles, rather than with the dynamics of the fluid alone. The research of how the surrounding fluid affects the dynamics of particles, or how to deal with the problem computationally for the microscopic level is still at the beginning. The aim of this study is to develop a simulation method where granular particles can be simulated inside fluids to study those problems. This is done by combining the simulation method for granular particles with the simulation method for the incompressible Newtonian fluid. The implementation of the algorithms which we will discuss is in two dimensions and developed using MATLAB [3].

1.1 Intended Characteristics of the Simulation

The intended region of validity for our approach is given in Fig. 1.1. The results in this thesis are computed for particles which are large than the size for which cohesion has to be taken into account. However, cohesion between particles could be implemented easily through modifications in the force laws between the particles. The flow is treated as isothermal flow as temperature effects in granular materials are usually absent or not measurable.

To reduce the amount of CPU-time the particles should be described by the minimal number of variables, so to obtain smooth shapes, we use a single polygon instead of a cluster of many circles. Further, the polygons allow a pore space around the particles which “fit” the particles when we use triangular meshes in order to

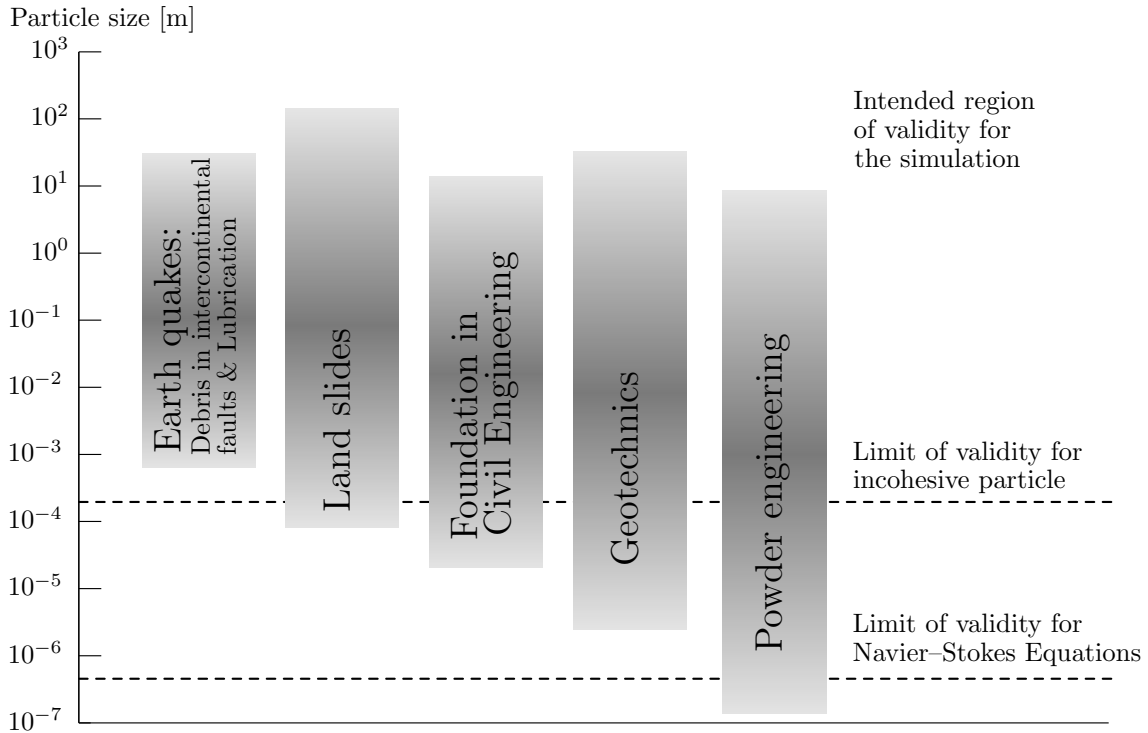


Fig. 1.1 Intended region of validity for our simulation of granular particles inside Newtonian fluid. The method should be applicable for technical problems as well as for systems in the geosciences.

minimize the noise in the solutions. As long as the simulation is stable, the largest possible mesh size should be allowed. An implicit time integrator is implemented in order to archive largest possible step-size.

As problems which deal with granular materials need the comparison with different simulation runs, running the simulation of each configurations on different personal computers (PCs) can be considered as a feasible “parallelization.” Therefore, the simulation should work on PCs and if possible make best use of multicore processors. Supercomputers as possible computing platforms are avoided because work overhead in the application procedures and and porting of the code (not to mention the risk of having the proposal rejected).

As many mechanisms of the dynamics of granular materials in fluid are still not clear, it is problematic to study the system in three dimensions as long as the understanding in two dimensions is still not established. Therefore, we limit ourselves to two dimensional in this study, where the granular particles can be imagined as rods or “Schneebeli materials.”

1.2 Microscopic and Macroscopic Simulation

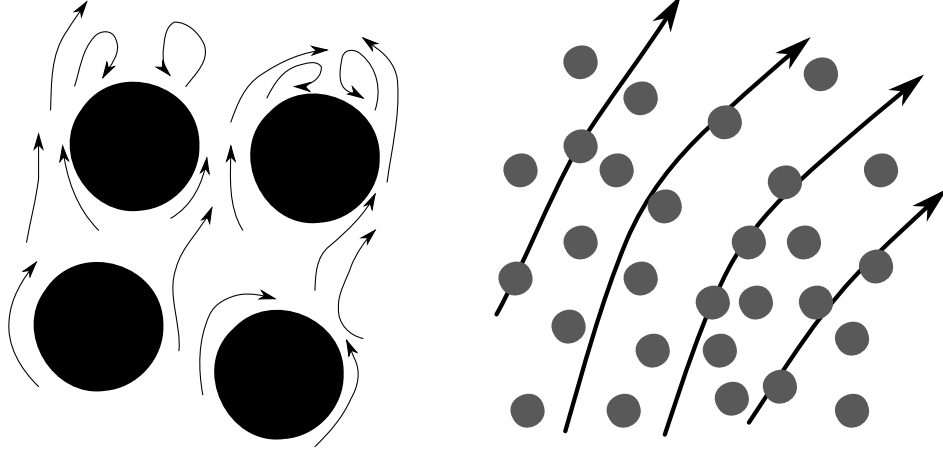


Fig. 1.2 Microscopic (left) modeling where the flow goes around the particles and macroscopic (right) modeling where the flow goes through the particles.

When simulating a particle-fluid system, from the modeling point of view, the methods can be categorized into microscopic or macroscopic simulations. In microscopic simulations as in our approach, the particles form exact boundaries of the surrounding fluid. In macroscopic simulations, the particles are not boundaries of the fluid, but overlay the fluid domain and experience forces from the underlying flow, which in some formulations is coupled back to the simulations as volume forces [4]. Such interactions are derived based on e.g. drag, but the problem is that there is no exact mathematical formulation of the drag of many particle configurations possible. One feature of “macroscopic” simulations is that the fluid will at least partially go “through” the solid particles. While for highly turbulent airflows and conditions where modeling is more important than exactness anyway, i.e. for milling of light small particles, the approach may have some justification. The higher the fluid density and the smaller the flow velocity is, the more dubious the approach becomes: Representing “exclusive volumes”, with the possibility to simulate blocking, is the most decisive interaction for slow flows, relevant for simulations from technical processes to disaster research. Because no simulation seems to exist which is satisfying both from the particle simulation point and the fluid mechanics point, we decided to design a microscopic simulation which is as realistic as possible for both flow and granular phase.

1.3 Simulation of the Particle Part

Depending on the application and the parameters, the granular particles which are treated in this study are sand grains, of the size of millimeters in diameter, up to fractured rocks, in the range of meters. The interaction between the particle surfaces in the normal direction of the contacts is due to elastic deformation, while in tangential direction, Coulomb friction plays a significant role. The deformations of the particles during contact are negligible relative to the displacements of the centers of mass. There is no universal governing equation for granular materials similar to the Navier–Stokes equations for the fluid. Modeling of granular materials based on continuum approaches usually includes a lot of phenomenological modeling and material constants with not clear physical meaning. For realistic simulations of granular materials we prefer the explicit modeling of the particle arrangement, i.e. solving the equations of motion of each individual particles directly.

Our simulation of the granular particles phase is carried out via the discrete element method (DEM) where the elastic contact force between two undeformed contacting particles is proportional to the overlapping area. The solid particles are treated as convex polygons instead of the conventionally used round particles, as the particle shapes play a very decisive role in constructing the aggregates. The whole granular dynamics is governed the competition between rolling and sliding, which crucially depends on the particle shape. With elastic contact force, dissipative force and tangential friction taken into consideration, the motion of each particles is obtained by solving the equations of motion using the integrator for ordinary differential equations, i.e. backward-difference formula of second order (BDF2).

1.4 Simulation of the Fluid Part

In principle, for a simulation of the fluid, one has the choice of either dealing with compressible or incompressible flow. For the case of fluid between granular particles, incompressible flow is the physically more valid situation: Compared to continuum materials, for granular materials from the same material, the sound velocity is reduced considerably: The sound velocities of an uncompressed granular packing is less than 10% for two-dimensional granular assemblies (rods) compared to the space-filling packing (homogeneous material) in simulations, and less than 1% for three-dimensional assemblies (plastic-beads) in experiments [5], as the transfer of momentum can take place via the relatively narrow particle interstices. As the sound velocity of the granular assembly will be much smaller than that of the fluid

we will treat the flow as incompressible. We have two possibilities of solving the equations of motion for the incompressible flow. For dealing with the incompressible flow, there is the alternative between Lagrangian methods (particle based, as for the granular particles), or Eulerian (mesh-based). We will shortly consider the possible choices.

1.4.1 Particle Methods for Fluid

The advantage of particle-methods is that they are already formulated in the “language” of the DEM-simulation, with forces, centers etc. instead of stresses, representative volumes etc. This makes the interaction with particles in principle “easy to implement”. In the meshless approach, the fluid volume is represented by a discrete number of particles. The solution of the flow problem i.e. velocities and pressures etc., obtained from the collective behavior of the particles which represent fluid volumes or parts of fluid volumes.

Combining Smoothed Particle Hydrodynamics with Discrete Element Method

Smoothed particle hydrodynamics (SPH) has originally been developed for astrophysics applications [6] for the exchange of mass between celestial bodies. Methods of coupling the SPH via implementation of the locally averaged Navier–Stokes equations with the DEM for the granular particles to simulate the fluid–particle systems has been presented in [7]. The method is completely particle-based for both particle- and fluid phase which avoids the usage of mesh. For the approach which simulates fluid with particle methods, the fluid boundary evolves on its own. Because the accuracy of the computation is proportional to the particle density, when rim frays due to scattering of SPH-particles on the border of granular particles, it is difficult to reduce the noise in the computation, except by massive use of particle, i.e. via CPU- and memory usage. A further drawback is that even stationary states have to be resolved with moving particles which might lead to “shot-noise” due to the motion of the discrete fluid particles. Another problem with obtaining stationary states is that it is unclear how long a particle simulation is supposed to run until the stationary state is reached: For mesh-based methods, obtaining the stationary flow is comparatively cheap. While SPH is inherently a “compressible” simulation (shocks can be resolved, which is relevant for astrophysical impact problems), the newer MPS (Moving Particle Semi-implicit [8]) an incompressible variant, which has

been used also in the coupling between “fluid” and “solid” particles. The problems are the same as for the SPH-approach, while straight surfaces of granular particles must be modeled by many “spherical” MPS-particles, which increases again CPU- and memory requirements.

Lattice Boltzmann Methods

While current lattice Boltzmann methods (LBM) [9] use amplitudes on a grid, the methods originated from cellular automata (lattice gas automata (LG) [10]) whose averaged properties obey the desired NavierStokes equation. Due to the original particle character, LBM is treated in this section. Lattice Boltzmann methods are constructed from simplified kinetic models that incorporate the essential physics of microscopic models with mesoscopic kinetic equations on continuous amplitudes so that for the macroscopic averaged properties the desired NavierStokes equations are recovered. LBM is very easy to use for modeling complicated boundary conditions and multiphase interfaces. To compute the fluid–solid problem, a solid particle is mapped onto the lattice to define boundary nodes [9]. A special rule at the boundary nodes is implemented to exchanges momentum between the fluid and solid particles at each update of the lattice. Fluid velocities at the nodes are matched to the particle velocities. Hydrodynamic forces and moments acting on solid particles are calculated from LBM and used to update the particle motion using Newton’s second law. One drawback of the lattice Boltzmann methods is the lack of Galilean invariance, after all, momentum conservation and the underlying Galilean invariance are the basic characteristics underlying the Navier–Stokes equation. Another problem is that lattice Boltzmann simulations tend to show very non-linear flow fields under conditions where it is difficult to verify whether the fingering etc. is due to the fluid mechanical problem or an artifact of the methods.

1.4.2 Grid Methods

The classical approach for simulating fluids is by spatial discretization of the flow equations onto grids: While the continuum equations have infinitely degrees of freedom, the purpose of the discretization is to yield a finite amount of equations and data.

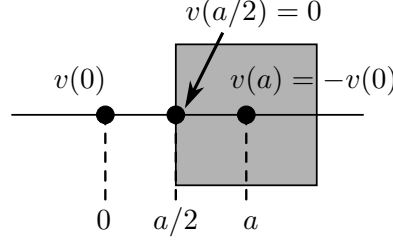


Fig. 1.3 Resolving the boundary $v(a/2)$ of a solid (in gray) by the specification of interpolated mirror condition $v(a)$ inside the solid.

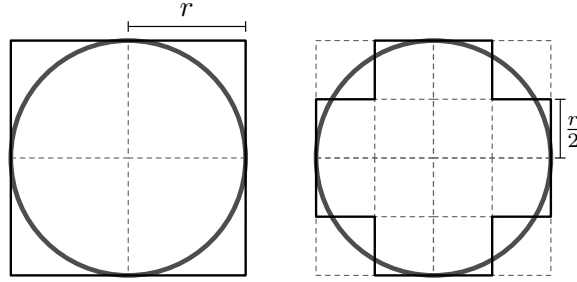


Fig. 1.4 Discretization of a circle via square grid: only the approximation of the area is improved with finer grid but not the circumference ($8r$ on the left, $16 \times \frac{r}{2} = 8r$ on the right).

Finite Difference Method

In finite difference (FD) methods, the original differential operators are approximated with difference formulae, in general in Cartesian coordinates. For pure fluid simulations and simple boundaries (parallel to the Cartesian axes), the use of fifth or higher order formulae is feasible and possible. For complicated boundaries, there are serious problems with higher order methods. Boundaries of solids must be resolved by the specification of interpolated mirror conditions inside the solid: If the boundary is at $a/2$ (see Fig. 1.3), in the middle of two grid points at 0 and a , where a is inside the solid, if $v(0)$ is computed, then $v(a)$ must be set as $v(a) = -v(0)$:

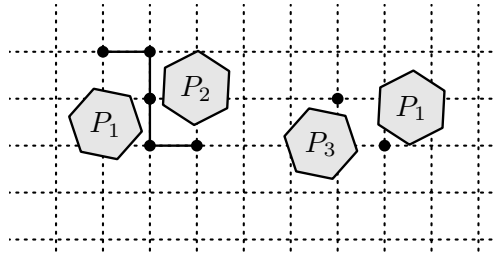


Fig. 1.5 Particle pairs P_1 – P_2 and P_3 – P_4 of the same size, relative position and orientation on finite difference square grid. Flow is possible for P_1 – P_2 but not for P_3 – P_4 .

This interpolates to $v(a/2) = 0$, no-flow conditions on the boundaries. For the boundaries at other spacing than at $a/2$, interpolations must be used. Nevertheless, such crude interpolations work only with low order methods, so that the commonly used approach is that of marker-and-cell (MAC) [4, 11, 2], in what is commonly called “second order”, but in fact is not [12]). If the boundaries are oblique to the coordinate axes (see Fig. 1.4), this means that curves of first or higher order must be approximated by grid points on rectangular grids: This is algebraically not possible, as points outside the Cartesian mesh points don’t enter the equations, so in principle any interpolation is arbitrary. We will return to this problem when we compare our approach with a FD approach in Chapter 5. Another problem related to the fact that FD-solutions are only defined on the grid points (“stencils”), not in between is the modeling of the connectivity of a pore space for particles a shorter distance (see Fig. 1.5): The flow will not depend on the inter-particle distance, but on the fact whether the line between two stencils is covered by one of the neighboring particles. The interpolation of the boundaries and the covering-uncovering of flow paths can be expected to lead to considerable noise in the simulation. Another issue is the pressure: While for FD-methods with compressible flow, the pressure fronts move along their own equations of (“sound”) motion, for incompressible flow the pressure should actually just “react” to the flow field: Nevertheless, the common approach is a relaxation of the pressure (as in MAC) to “physically plausible” values which are nevertheless not physical, as the relaxation equations are not motivated by physical necessity but by numerical convenience. To summarize: The need to model the particle surfaces via interpolated boundary conditions allows only the use of low order solvers, which produce noisy solutions, the use of grids leads to pore spaces which increase the noise further, and the relaxation procedures used in this context are not exact either. All this leads to noisy solutions which must be obtained with small grids and small time-steps, which is numerically not feasible. The resulting simulations might work for only some specific problems and some unphysical artifact can be observed. In [13] where the microscopic approach is via direct numerical simulation, the particles showed a suspicious lack of settling. The three dimensional simulation with ellipsoidal particles in [4] (which is based on [2] but macroscopic instead of microscopic) had “springs” fixed between particles and fluid. These springs were sometimes unintentionally fixed on the boundary due to rounding errors in the position assignment of particles and boundaries, which then lead to particles sticking on the boundaries. Both Ristow [2] and Schwarzer [4] reportedly withdrew from academic research after completion of their code. The student who inherited the code from [4] dropped PhD as the program crashed

continuously after 10 hours of CPU-time. So completing a working simulation code for fluids and particles one can have confidence in by no means a trivial affair.

Finite Volume Codes

Finite volume codes have been used to model particles in fluids [14]. A drawback with finite volume methods is that they need grids with well-defined connectivities, while the connectivity of the pore space for granular particles in fluids can be expected to change in every time-step, which is one reason that we have avoided this discretization approach. Another reason is that finite element theory allows to classify finite volume methods as a kind of finite element methods where the some subset of equations remains unsolved, but the solutions are set.

Eulerian Continuum Approach

The Eulerian continuum approach uses continuum theory that views solid and fluid as inter-penetrating mixtures [15]. This approach derives the disperse-phase momentum equation by averaging the particle equation of motion. Interactions between the solid and the fluid cannot be understood from these mixture theories alone. Effects from particle collision also have been neglected in the study.

1.4.3 Finite Element Methods

As we want to simulate polygonal particles, the resulting pore space can be discretized exactly only via the use of triangular grids: Every other spatial discretization is inappropriate, noise resulting from approximating oblique boundaries with rectangular grids leads to noise which has unpredictable effects on the stability and physicality of the simulation. We choose to simulate the fluid phase using the finite element method (FEM) (a Galerkin method with piecewise polynomials) as it allows the usage of triangular meshes. Another kind of Galerkin methods are spectral methods. As the underlying functions are periodic, spectral methods have ideal momentum-conservation properties, as they are formulated in Fourier-space. However, for systems with many particles, the resolution of the boundaries will be a serious problem, as it would be necessary to use the Fourier-transform forward and backward to calculate the interaction between particles and flow. Our implementation of the Navier-Stokes equations for incompressible flow via the Galerkin finite element method (FEM) is formulated as differential algebraic

equations (DAE) with the pressures as the Lagrange parameters. The time integration is again via the backward-difference formula of second order (BDF2) while the resulting non-linear equations are solved by Newton–Raphson methods. The grid is obtained for Taylor–Hood elements from Delaunay triangulations with additional post-processing steps. Form drags and friction drags from the fluid acting on the particles are computed from the velocities and pressures obtained via the finite element method. Boundary conditions for the flow around the particle boundaries are determined by the velocities of the solid particles.

Not all finite element approaches are feasible in combination with particle simulations. Simulation of the particle–fluid system via stabilized space-time finite element method have been reported in [16]. In that method both temporal coordinate and the spatial coordinates are discretized using the finite element method. The deformation of the spatial domain with time is reflected in the deformation of the mesh. Linear interpolation functions were used for both fluid velocities and pressures in the study. The problem with this approach is that the behavior of finite element methods in the time domain is hardly understood: It is not among the standard methods [17] for which decades of experience with various problems exist. This makes the approach a bit risky, as there is a possibility of bad surprises.

1.5 Performance Issues

Though our approach is computationally costly, the amount of available CPU-power from PC’s and servers is still increasing exponentially. It seems that substantial sustained computing power has become available in the hands of researchers who are far away from supercomputing centers with their cumbersome application procedures. Our intention is, rather than focusing on massive-parallel implementations with time-consuming parallelizations, to develop a methodology where we can reduce the degrees of freedom without lowering the accuracy of the geometrical description, and to use time-integration methods for which the time-step is only limited by the time-scale of the physical phenomena, rather than by the lamentable stability requirements of easy-to-implement algorithms. Once we have a satisfying algorithm, we can focus on the performance optimization.

1.6 Overview of the Thesis

This thesis discusses the necessary formulations and algorithms for both the granular particle phase (Chapter 2) and the fluid phase (Chapter 3). Then we demonstrate the capability of our FEM-code through introducing a novel technique for simulating free surfaces and its verification in Chapter 4. The coupling of the discrete element method for the granular particles and the finite element method for the fluid is discussed in Chapter 5: The verification of the DEM-FEM approach is via the computation of wall correction factors of a sinking particle. Chapter 6 discusses two numerical applications which were conducted with the DEM-FEM code: compaction due to tapping in two-dimensional granular columns and collapse of granular column in fluid. Finally, the limitations and possible improvements of the simulation is addressed in Chapter 7, and the summary of the development of the DEM-FEM code is given in Chapter 8.

Chapter 2

Simulation of the Granular Part

2.1 The Principle of Discrete Element Method

The discrete element simulation provides a way to study and analyze the micromechanics of granular material in a way that cannot be achieved with the continuum approaches. The method treats granular materials as assemblies of particles and the motion and interaction between individual particles are computed throughout the simulation. Forces which are taken into consideration when computing the motion of particles include

- the elastic contact force in normal direction,
- the dissipative force in normal direction,
- friction in tangential direction.

Before we go into the detailed discussion of the magnitude of the forces mentioned above, we need to define the direction and the contact point of the forces. The definitions for both the magnitudes, the directions of the forces and the force point need to be “robust” because the time integrator (see Section 2.3) assumes continuity in the forces. By robust we mean that continuous relative position changes between contacting particles will lead to continuous changes in the force (both magnitude and direction) and the force point. The simulation will be unstable if a small change in position can cause a discontinuous change in the force (be it in magnitude or direction). Because the flow simulation in this thesis is limited to the two-dimensional case, we also limit our discussion of the discrete element method to the two-dimensional case.

In the simplest case of the contact of a pair of circular particles the straight line $\overline{C_1C_2}$ connecting two centers of mass (C_1 and C_2 in Fig. 2.1) is used as the normal direction \hat{n} and the tangential direction \hat{t} can be obtained by taking the direction orthogonal to the normal direction. The force point P is defined at the intersection between $\overline{C_1C_2}$ and the line $\overline{S_1S_2}$ connecting the two intersection points S_1 and S_2 . This location can be obtained easily as the distance from C_1 ,

$$\overline{C_1P} = \frac{R_1^2 - R_2^2 + \overline{C_1C_2}^2}{2\overline{C_1C_2}} \quad (2.1)$$

using the radii of the particles R_1 and R_2 without the need of computing the actual intersection points S_1 and S_2 . In order to obtain consistent directions throughout the simulation, the tangential direction is defined with respect to the counterclockwise orientation of the first particle of the contacting pair, i.e. direction of \hat{t} is $\overrightarrow{S_2S_1}$ (not $\overrightarrow{S_1S_2}$). Fig. 2.1 is determined by the counterclockwise orientation of particle at C_1 . The orientation of the normal direction can then be defined as $\hat{n} = [-\hat{t}_y \ \hat{t}_x]^\top$, i.e. counterclockwise rotation of \hat{t} by 90 degrees. Definitions of force directions and force point for non-round particles will be introduced in Section 2.2.4. For the sake of clarity, we will use \perp and \parallel in the following equations to indicate the normal direction (for elastic force and damping) and tangential direction (for friction) respectively. For circular particles, the normal forces act along the line defined by the centers of mass, they are central forces, while for elongated particles, this is usually not the case.

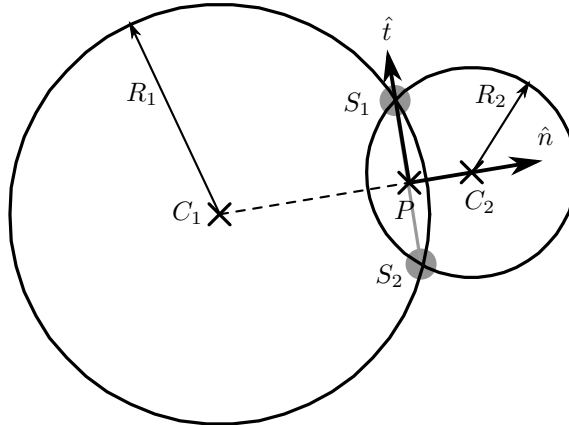


Fig. 2.1 The normal \hat{n} and tangential \hat{t} directions of two contacting round particles at C_1 and C_2 with radii R_1 and R_2 respectively. The force point P is located at the intersection between $\overline{C_1C_2}$ and $\overline{S_1S_2}$.

2.1.1 Elastic Force in Normal Direction

Elasticity theory yields the following behavior for the elastic contact force between two contacting particles (due to symmetry argument, the proportionality is the same for equally shaped contacts or contacts of the same shape with a plane)[18]

- Hookean (linear) contact where the elastic force is proportional to δ for rectangular shape particles (see (a) in Fig. 2.2).
- Hertzian contact where the elastic force is proportional to $\delta^{3/2}$ for circular shape particles (see (b) in Fig. 2.2).
- Wedge-shaped contact where the elastic force is proportional to δ^2 for particles with sharp corners (see (c) in Fig. 2.2).

The overlap area of the contacts are also (at least approximately) proportional to the penetration depth δ [19]. Therefore, choosing the magnitude of the elastic force to be proportional to the overlapping area A instead of the penetration depth δ is more practical than working with the penetration depth because we can reproduce all three regimes mentioned above without the need to analyze the contact types and apply the right exponents of δ for the corresponding geometries. The shape of the particle is treated as invariant, i.e. there will not be any deformation due to the contact. The magnitude of the overlap relative to the particle size will be small. The overlap of the particles can be interpreted as the the necessary amount of deformation which is required so that the particles can occupy the space in the actual configuration. This finite overlap approach which is also know as “hard particle, soft contact” saves us the effort of detecting a exact point contact with the precision necessary for rigid particle simulations and model the deformation, as would be necessary in a finite element simulation.

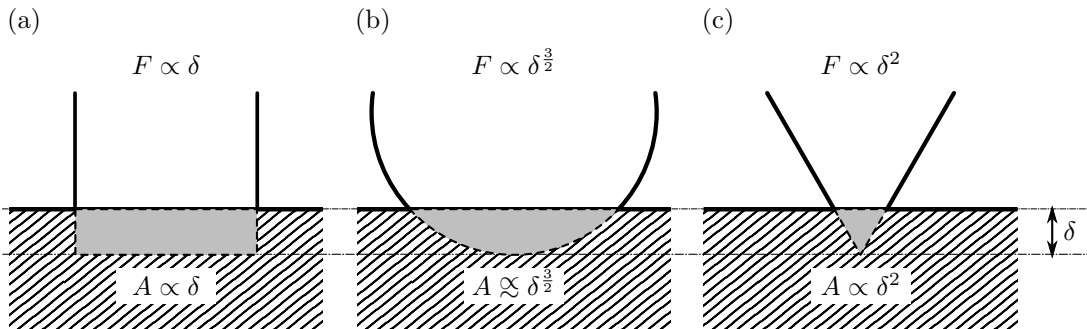


Fig. 2.2 Relationships between the elastic force F , contact area A , and penetration depth δ for three types of contact based on the elasticity theory: (a) Hookean contact, (b) Hertzian contact (middle), and (c) wedge-shaped contact.

Another parameter which is used to compute the elastic contact force is the Young's modulus Y . In the case of two dimensions where in our case the particles are conceived as 1 [m]-long rods, the Young's modulus is rescaled from the three dimensional units $[\text{N}/\text{m}^2]$ to two dimensions $[\text{N}/\text{m}]$. Other constants such as Poisson's ratio, bulk modulus, etc. are neglected due to the fact that for the inhomogeneous material composition of grains, no continuum assumptions are valid, and the effects due to the corresponding material parameters are weaker anyway.

However, multiplying the two-dimensional Young's modulus Y $[\text{N}/\text{m}]$ with the area A $[\text{m}^2]$ does not yield a force, as we are still missing a factor with a unit of length. The additional factor must be supplied by a different reasoning. Sound velocity c $[\text{m}/\text{s}]$

$$c = \sqrt{\frac{Y}{\sigma}} \quad (2.2)$$

depends only on the (in our case, two-dimensional) density σ $[\text{kg}/\text{m}^2]$ and the Young's modulus Y . Therefore its value should be the same for a bulk continuum and a space-filling packing of particles regardless of the particle-size. Sound propagation occurs in discrete element simulations through the microscopic displacements of the particles' centers of mass [5, 20]. A "characteristic length" l_c is added to supply the missing factor of length in the formulation of elastic force

$$l_c = \frac{r_1 r_2}{r_1 + r_2} \quad (2.3)$$

in order to obtain sound velocity which is independent of the particle size. Here $r_1 = \|\mathbf{r}_1\|$ and $r_2 = \|\mathbf{r}_2\|$ are the distances from the center of mass to the force point P for each particle (see Fig. 2.3).

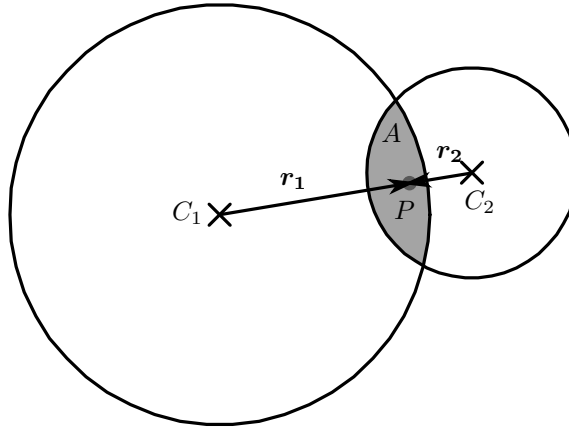


Fig. 2.3 Overlapping particle-pair with overlap area A ; r_1 and r_2 are distances from the centers of mass (C_1 and C_2) to the force point P .

Thus, using the two-dimensional Young's modulus Y , the overlap area A and the characteristic length l_c which serves to adapt the sound velocity in a space-filling packing to the sound velocity of the material, the magnitude of our shape-dependent elastic force $F_{c,\perp}$ of two contacting particles can be written as

$$F_{c,\perp} = Y \frac{A}{l_c}. \quad (2.4)$$

2.1.2 Dissipative/Damping Force in Normal Direction

The deformation rate at the contact between two particles gives rise to a normal viscous force. The damped harmonic oscillator

$$m \frac{d^2x}{dt^2} = -kx - c \frac{dx}{dt} \quad (2.5)$$

with spring constant k [N/m] its magnitude is determined by the viscous damping coefficient c [Ns/m]. Using the dimensionless damping constant $\gamma = c/\sqrt{km}$, the viscous term can be rewritten as

$$-\gamma \sqrt{km} \frac{dx}{dt}. \quad (2.6)$$

For our force law Eq. (2.4) we adapt this dependency by replacing the position x with area A , and the spring constant k with the Young's modulus Y as in the equation for the elastic force (Eq. (2.4)), then we use the characteristic length l_c to "correct" the unit. Then the resulting dissipative force $F_{d,\perp}$ can be written as

$$F_{d,\perp} = \gamma \sqrt{Y m_{\text{red}}} \frac{1}{l_c} \cdot \frac{dA}{dt}, \quad (2.7)$$

with the force proportional to the rate of change of the overlapping area $\frac{dA}{dt}$ as the viscous damping term in the harmonic oscillator proportional to the velocity. The reduced mass m_{red} for the masses m_1 and m_2 of the two contacting particles

$$m_{\text{red}} = \frac{m_1 m_2}{m_1 + m_2} \quad (2.8)$$

is included so that the force law can deal with both particle-particle and particle-wall interactions. When a particle with mass m_1 collides with a large wall ($m_2 \rightarrow \infty$) (treated as a particle which is fixed throughout the simulation) the obtained reduced mass m_{red} will be close to m_1 . The prefactor $\sqrt{Y m_{\text{red}}}$ scales the dissipation for different various masses m and Young's moduli Y . When applying the formulation

for the viscous damping into the code, Eq. (2.7) can be rewritten as

$$F_{d,\perp} = \gamma \sqrt{Y m_{\text{red}}} \frac{1}{l_c} \cdot \frac{A(t) - A(t - \tau)}{\tau}, \quad (2.9)$$

where τ is the time-step and $A(t - \tau)$ is the overlap area of the particle pairs in the previous time-step.

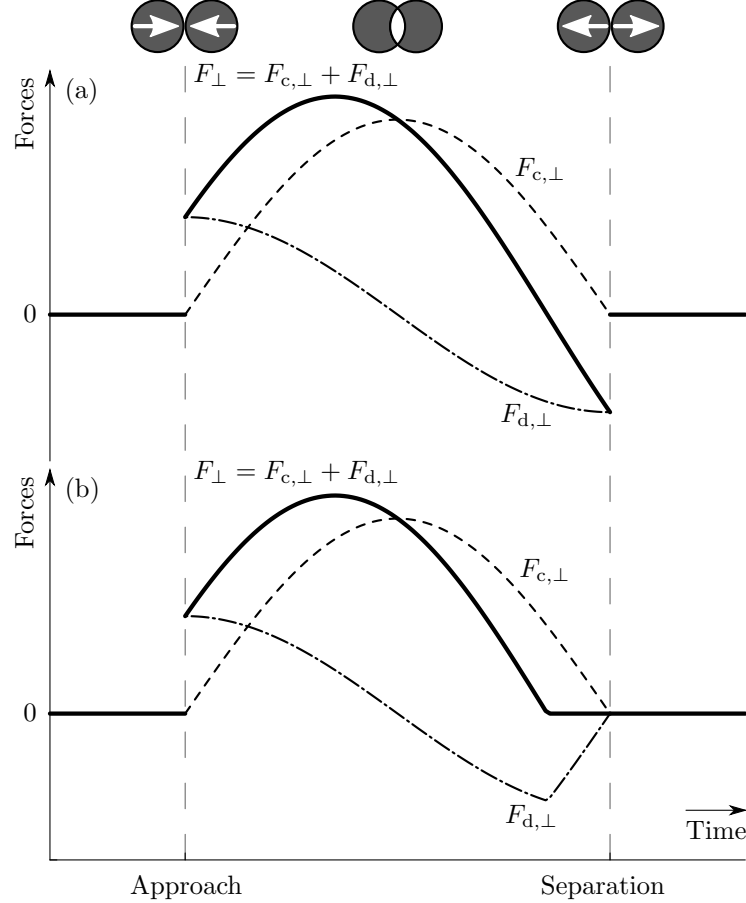


Fig. 2.4 (a) Direct summation of the elastic $F_{c,\perp}$ and damping force $F_{d,\perp}$ in normal direction leads to jumps in total force F_\perp during the approach and separation of the contacting particles. (b) Correction in the damping force is performed to cut off the unphysical attractive force during separation.

However, following the example of the harmonic oscillator in the above way leads to problems for simulations with closing and separating contact. The resulting total force from the summation of the elastic contact force $F_{c,\perp}$ and the dissipative force $F_{d,\perp}$ will give unphysical spurious attractive forces. The reason is the non-smooth evolution of the dissipative force during closing and opening of the interparticle contact. This can be understood by observing that during a collision, the elastic force evolves proportional to a sine curve (proportional to the overlapping area) and the the dissipative force evolves proportional to a cosine curve (proportional to

the first derivative of the overlapping area) over the interval between 0 and π as shown in Fig. 2.4. Jumps occur for the cosine at 0 and π , due to the fact that the velocities are maximal during closing of the contact and during separation, where the elastic force is negligible. Accordingly, these jumps occur also the total the total interparticle force. Physically, since impacts are non-smooth processes which trigger sound and damage at the surface, the jump at approach can be justified and the backward differentiation formula (see Section 2.3) can deal with it without any treatment if the impact velocity is not too large. However, the attractive force at separation is unphysical since the total force for dry granular materials without any additional cohesive force can only be repulsive or zero. In order to cut off the unphysical attractive force, the following modification is applied in the dissipative force when the total force has become “attractive”

$$F_{d,\perp} = \begin{cases} -F_{c,\perp} & \text{if } F_{c,\perp} (F_{c,\perp} + F_{d,\perp}) < 0, \\ \gamma \sqrt{Y m_{\text{red}}} \frac{1}{l_c} \cdot \frac{dA}{dt} & \text{otherwise.} \end{cases} \quad (2.10)$$

If this cutoff is not implemented, unphysical noise is introduced in the simulation and the finite difference schemes used for the time-integration of the particles become unstable. In combination with the fluid code, the noisy movement of the particles cannot be damped out by the fluid, because the motion amplitudes are unphysically large. On the contrary, the noise is transferred to the flow field, so that the fluid part may blow up first.

2.1.3 Friction in Tangential Direction

The dynamic Coulomb friction F_{dy} of one-dimensional dry sliding contacts between two solids can be written as

$$F_{dy} = -F_n \mu \operatorname{sgn}(v), \quad (2.11)$$

where F_n is the normal force and μ is the friction coefficient. F_{dy} is a dissipative force tangential to the normal contact and acts in the opposite direction to the sliding (tangential) velocity. On the other hand, the static Coulomb friction F_{st} for resting contacts is given by the inequality

$$-F_n \mu \leq F_{st} \leq F_n \mu \quad (2.12)$$

which is a constraint force without any dissipation at all (see Fig. 2.5). In general, it is justified to use the same friction coefficient μ for both dynamic and static frictions for contacts which do not change chemically or mechanically over time [19].

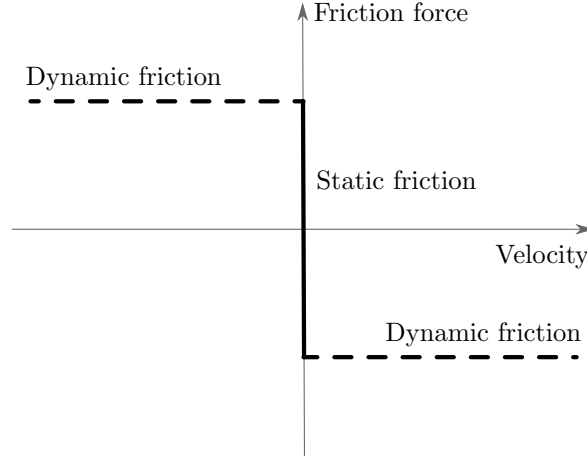


Fig. 2.5 Two kinds of friction in sliding contact of particles: static friction as constraint force and dynamic friction as dissipative force.

The friction coefficient is independent of the material strength and the apparent contact area. It is not much affected by the surface roughness either. The friction coefficient is mainly a result of the reaction of the “unemployed” surface electrons and the electron affinity (the likelihood for bonding between electrons) of the two contacting solids which contribute to the adhesion. A common example would be the low friction coefficient between Teflon and metals. The difference between energy states (metal bonding for metal and covalent bonding for Teflon) results in poor electron affinity and leads to a low friction coefficient. Accordingly, friction is different from interlocking of surfaces, which is a geometrical effect and must be modeled via the shape of the particles.

Since until today there is no exact computation method for static friction in general many-particle systems available, the friction force in our discrete element simulation is computed from the Cundal–Strack model [21]. The tangential friction force $F_{d,\parallel}$ at time t is modeled incrementally from the previous time-step τ with in the direction opposite to the relative tangential velocity $v_{\text{rel},\parallel}$ at the contact

$$F_{d,\parallel}(t) = \begin{cases} \text{sgn}(F_{d,\parallel}(t)) \cdot \mu F_{\perp} & \text{if } F_{d,\parallel}(t) > \mu F_{\perp} \\ F_{d,\parallel}(t - \tau) - Y_{\parallel} v_{\text{rel},\parallel} \tau & \text{otherwise.} \end{cases} \quad (2.13)$$

where $Y_{\parallel} = \frac{2}{7}Y$ is the “tangential stiffness” with the same dimension as a spring constant [N/m] and $F_{\perp} = F_{c,\perp} - F_{d,\perp}$ is the total force in normal direction. The

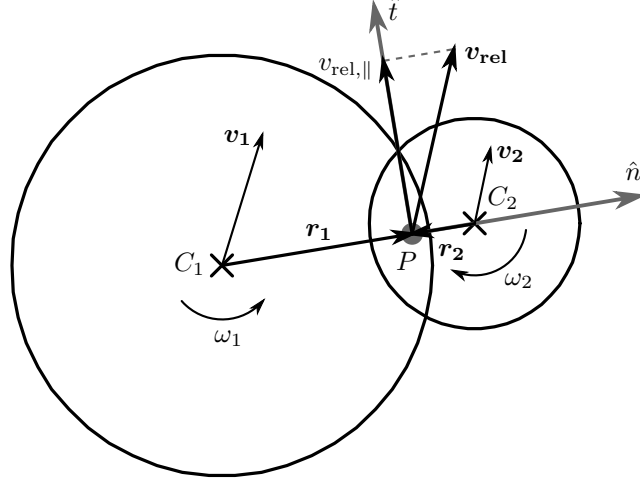


Fig. 2.6 Obtaining the relative tangential velocity $v_{\text{rel},||}$ from velocities \mathbf{v}_1 , \mathbf{v}_2 , angular velocities ω_1 , ω_2 of the contacting particles and the vectors \mathbf{r}_1 , \mathbf{r}_2 between their centroids and the force point P .

relative velocity \mathbf{v}_{rel} is derived from the velocities \mathbf{v}_1 , \mathbf{v}_2 of contacting particles, angular velocities ω_1 , ω_2 around their centers of mass, and the vectors \mathbf{r}_1 , \mathbf{r}_2 between the centroids of the particles and the force point P (see Fig. 2.6)

$$\mathbf{v}_{\text{rel}} = \mathbf{v}_1 - \mathbf{v}_2 - (\omega_1 \mathbf{r}_1 - \omega_2 \mathbf{r}_2). \quad (2.14)$$

The relative tangential velocity $v_{\text{rel},||}$ in Eq. (2.13) can be obtained from the projection of the relative velocity onto the tangential direction.

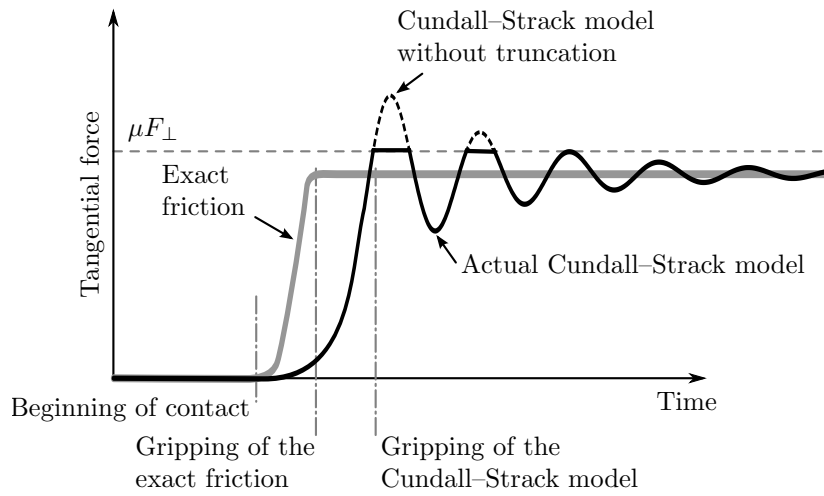


Fig. 2.7 Implementation of Cundall-Strack model for the tangential friction force in the discrete element simulation. The model is truncated to the Coulomb friction μF_{\perp} . Gripping is delayed in the Cundall-Strack model comparing to the exact friction.

The upper case in Eq. (2.13) means that if the obtained incrementing tangential force is larger than the maximal value allowed for dynamic friction, it will be truncated. This approach is sometimes known as a model of “breaking tangential spring”. Dividing the lower part of Eq. (2.13) with an infinitesimal time-step τ gives the following differential equation

$$\frac{dF_{d,\parallel}(t)}{dt} = -Y_{\parallel}v_{\text{rel},\parallel}. \quad (2.15)$$

where the tangential force will not be increasing monotonously but has a behavior of a harmonic oscillator. This means that the tangential force does not always act opposite to the tangential velocity. The energy is dissipated only if the tangential force reaches the value for sliding friction. The oscillations can be reduced by introducing a damping term proportional to $\sqrt{m_{\text{eff},\parallel} \cdot Y_{\parallel}}$ so that we have

$$F_{d,\parallel}(t) = \begin{cases} \text{sgn}(F_{d,\parallel}(t)) \cdot \mu F_{\perp}, & \text{if } F_{d,\parallel}(t) > \mu F_{\perp} \\ F_{d,\parallel}(t - \tau) - Y_{\parallel}v_{\text{rel},\parallel}\tau - \sqrt{m_{\text{eff},\parallel} \cdot Y_{\parallel}}v_{\text{rel},\parallel}, & \text{otherwise,} \end{cases} \quad (2.16)$$

where the tangential mass $m_{\text{eff},\parallel}$ is defined from the particles’ mass (m_1 and m_2) and their distances from the centers of mass to the force point (r_1 and r_2 in Fig. 2.3) as

$$m_{\text{eff},\parallel} = \frac{1}{\frac{1}{m_1} + \frac{1}{m_2} + \frac{r_1^2}{I_1} + \frac{r_2^2}{I_2}} \quad (2.17)$$

where the momenta of inertia I_1, I_2 being included in the reduced “tangential” mass.

Beside the oscillatory behavior, the Cundall–Strack model also leads the delay in gripping of the tangential friction compared to the “exact” friction. Nevertheless, we can still obtain good results (e.g. constructing a stable heap on a smooth surface) as beyond the time-scale of the oscillation is considerably shorter than the time-scale of the physical processes under consideration. For particles in fluids, the oscillations are suppressed anyway, and the damping term in Eq. (2.16) is not necessary.

2.1.4 Total Force and Torque

The resulting normal and tangential forces from the sections above need to be decomposed into their horizontal and vertical components (see Fig. 2.8) before

summing the forces together to get the total force

$$\mathbf{F} = \begin{bmatrix} F_x \\ F_y \end{bmatrix} = (F_{c,\perp} - F_{d,\perp}) \begin{bmatrix} \hat{n}_x \\ \hat{n}_y \end{bmatrix} + F_{d,\parallel} \begin{bmatrix} -\hat{n}_y \\ \hat{n}_x \end{bmatrix}. \quad (2.18)$$

The torques \mathbf{T}_1 , \mathbf{T}_2 acting on particle 1 and 2 can then be computed as

$$\begin{aligned} \mathbf{T}_1 &= \mathbf{r}_1 \times \mathbf{F}, \\ \mathbf{T}_2 &= \mathbf{r}_2 \times -\mathbf{F}. \end{aligned} \quad (2.19)$$

In the case of particles of different shape and size, there is no action = reaction principle, i.e. in general $\mathbf{T}_1 \neq \mathbf{T}_2$.

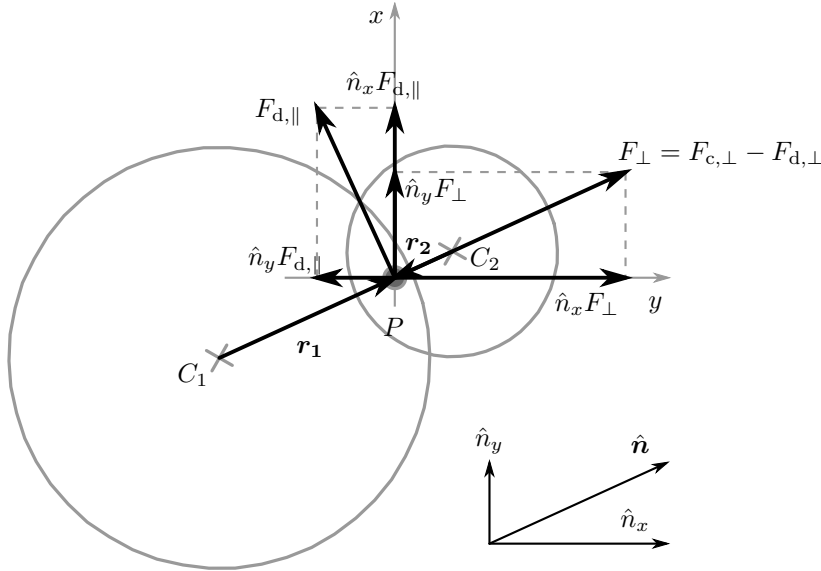


Fig. 2.8 Decomposition of the normal forces F_\perp (elastic $F_{c,\perp}$ and damping $F_{d,\perp}$ forces) and tangential friction force $F_{d,\parallel}$ between two contacting particles into horizontal and vertical components.

2.2 Polygonal Particles

A very tempting approach in the discrete element simulations is to use round particles for simplicity. Unfortunately, particle shapes do play a very decisive role in constructing the aggregates. For example, non-elongated particles have angles of repose of about 20 degrees, while round particles do not even give straight angles of repose. So the use of round particles limits the phenomena which are accessible with discrete element simulations considerably.

The dynamics of the granular materials is basically determined by the competition between the rolling and sliding. Unlike round particle, a non-round particle will always need a finite torque in order to roll due to the finite length of their edges. If the rolling is inhibited, the dynamics will then be determined by sliding alone. Round particles can escape the influence of sliding friction at zero energetic cost by rolling.¹ Therefore, systems which are made up of round particles will be unstable due to the fact that the particles have very high tendency to roll with relatively small mechanical resistance and low energetic cost. The competition between rolling and sliding also effects the strength of a granular assembly (as measured by e.g. triaxial compression). The strength of an assembly with round particles can be expected to be weaker than the one with convex particles. The assembly with non-convex particles will have higher strength than the other two due to the higher degree of interlocking of surfaces. In order to “mimic” the behavior of the non-round particles assembly in a round particles simulation, in some simulation packages for round particles unphysically large rolling friction coefficients are specified to enforce higher angles of repose. However, using the high rolling friction coefficient which can never be archived in experiment does not improve the verisimilitude of the simulation.

Another aspect is that the pore space which will contain the fluid simulation is difficult to discretize if the wall boundaries are not straight. In this respect, the use of polygonal particles is also advantageous because the resulting polygonal porespace can be discretized exactly at least by finite element methods. In the following sections, we will discuss the basic concepts in using the full geometric information of the contacting polygonal particles to model the elastic contact force discussed in Section 2.1.1. These include the computation of the overlap area A in Eq. (2.4) and the necessary modifications in the force directions and force points.

2.2.1 Creating the Polygonal Particles

In the following we will introduce two methods that are used in the program to generate convex polygonal particles for the simulations. To simplify the algorithms for overlap computations (will be introduced in Section 2.2.3) it is advantageous that the edges and vertices are labeled counterclockwise.

¹Experimentally, rolling coefficients of friction are two to three orders of magnitude smaller than sliding coefficients of frictions, so no coefficient of rolling friction is used in the program.

The first method inscribes a regular polygon into an ellipse (with semi-major axis a and semi-minor axis b)

$$\begin{aligned} x_i &= a \cos \theta_i, \\ y_i &= b \sin \theta_i. \end{aligned} \tag{2.20}$$

where θ_i can be chosen at certain regular stepping, $i = 1, 2, \dots, n$ indicating the i th of n vertices, (x_i, y_i) are the coordinates of the vertices (see Fig. 2.9 (a)). Additionally, a small dispersion can be added in the half-axes a , b and the angle θ with random numbers in order to generate an irregular convex polygon (gray in Fig. 2.9 (a)). After the randomization, the convexity of the polygon should be verified, as too large random amplitudes can make the particles non-convex, in which case the overlap computation and the force computation will fail.

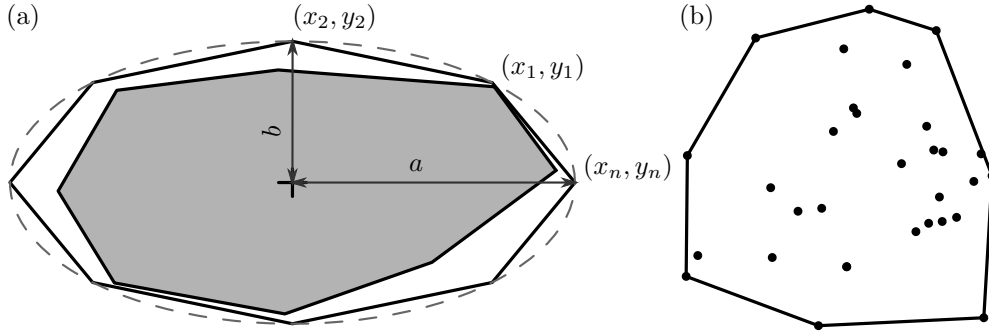


Fig. 2.9 (a) Constructing a polygon by inscribing a regular polygon with eight vertices (solid line) into an ellipse (dashed-line) (with semi-major axis a and semi-minor axis b). Irregular polygon (gray) can be obtained via randomization of the vertices of the regular polygon. (b) Constructing a polygon via convex hull for a set of random points (\bullet). The polygons obtained with this method has a higher tendency to have sharp corners.

Computing the convex hull is another method of creating a polygonal particle for a set of random points (see Fig. 2.9 (b)). Computation of the convex hull is implemented in MATLAB with `convhull(x, y, 'simplify', true)` where x and y are the coordinates of the random points. However, this method tends to create sharp polygonal shapes for few points and rectangular corners for many points which then lead to penetrating contacts which will be dealt with in section 2.2.5.

2.2.2 Mass, Center of Mass and Moment of Inertia

Since computing mass, center of mass and moment of inertia of a triangle is easy, the computation of these quantities for a convex polygon is done by splitting the

polygon into triangles. It is convenient to select a point \mathbf{P} inside the polygon as the average of all n vertices of the particle (see Fig. 2.10 (a)) $\mathbf{x}_i = [x_i \ y_i]^\top$

$$\mathbf{P} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i. \quad (2.21)$$

From \mathbf{P} and \mathbf{x}_i we can obtain the vectors \mathbf{r}_i in radial direction for each vertex

$$\mathbf{r}_i = \mathbf{x}_i - \mathbf{P} \quad (2.22)$$

which partitions the polygon into n triangles. The vectors \mathbf{s}_i which describe the edges of the particle are given by the coordinates of the i th and $(i+1)$ th vertices

$$\mathbf{s}_i = \mathbf{x}_{i+1} - \mathbf{x}_i. \quad (2.23)$$

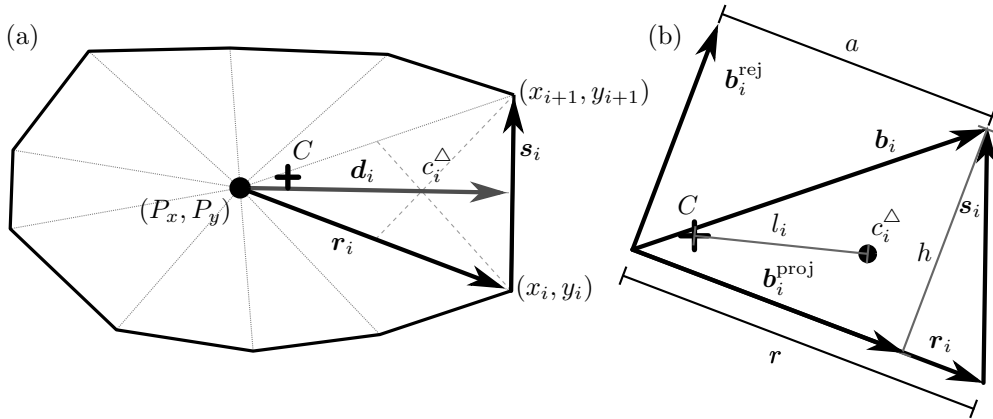


Fig. 2.10 (a) Decomposition of a particle with $n = 10$ vertices into n triangles. The center of mass of the i th triangle can be obtained from the median as $\frac{2}{3}\mathbf{d}$. The center of mass of the polygon \mathbf{C} will be the weighted average of the centers of mass of the triangles. (b) Triangle enlarged from (a) to show the computation of the moment of inertia using the radial vector \mathbf{r}_i and edge vector \mathbf{s}_i .

Mass

The area A_i of the triangular partition formed by radial vector \mathbf{r}_i and edge vector \mathbf{s}_i is given by the cross product

$$A_i = \frac{1}{2} \|\mathbf{r}_i \times \mathbf{s}_i\|. \quad (2.24)$$

Multiplying the area by the two-dimensional density gives us the mass of the triangle

$$m_i^\Delta = \sigma A_i, \quad (2.25)$$

and the mass of the particle m is simply the sum of mass of all triangular partition

$$m = \sigma \sum_{i=1}^n A_i. \quad (2.26)$$

Center of Mass

The center of mass of a triangle is located at the intersection of the medians, i.e. the lines joining each vertex with the midpoint of the opposite edge. The center of mass divides the median in the ratio 2 : 1. The vector \mathbf{d}_i from point P to the midpoint of \mathbf{s}_i will be one of the medians which can be written as the addition of the radial and edge vectors

$$\mathbf{d}_i = \mathbf{r}_i + \frac{1}{2}\mathbf{s}_i. \quad (2.27)$$

The center of mass of the each triangle can be computed as the following based on the ratio mentioned above

$$\mathbf{c}_i^\Delta = \mathbf{P} + \frac{2}{3}\mathbf{d}_i = \mathbf{P} + \frac{2}{3}\left(\mathbf{r}_i + \frac{1}{2}\mathbf{s}_i\right). \quad (2.28)$$

The center of mass of the polygon \mathbf{C} will be the weighted average of the centers of mass of the triangles

$$\mathbf{C} = \frac{\sum_{i=1}^n m_i^\Delta \mathbf{c}_i^\Delta}{m} \quad (2.29)$$

which is close to but not identical to the average of the vertices \mathbf{P} .

Moment of Inertia

The moment of inertia of a triangle around the axis through its center of mass is given by

$$I^\Delta = \frac{r^3 h - r^2 h a + r h a^2 + r h^3}{36} \quad (2.30)$$

where for the i th triangle of the polygon the base is given by

$$r = \|\mathbf{r}_i\|. \quad (2.31)$$

The vector \mathbf{b}_i from the endpoint of the base to the apex (see Fig. 2.10 (b)) can be obtained from \mathbf{r}_i and \mathbf{s}_i

$$\mathbf{b}_i = \mathbf{r}_i + \mathbf{s}_i, \quad (2.32)$$

from which the length a can be obtained from the norm of its projection onto \mathbf{r}_i

$$\mathbf{b}_i^{\text{proj}} = \frac{\mathbf{b}_i \cdot \mathbf{r}_i}{\|\mathbf{r}_i\|^2} \mathbf{r}_i, \quad (2.33)$$

$$a = \|\mathbf{b}_i^{\text{proj}}\|. \quad (2.34)$$

On the other hand, the height h is equivalent to that component of \mathbf{b} which is orthogonal to \mathbf{r}_i (i.e. the rejection of \mathbf{b}_i onto \mathbf{r}_i)

$$\mathbf{b}_i^{\text{rej}} = \mathbf{b}_i - \mathbf{b}_i^{\text{proj}}, \quad (2.35)$$

$$h = \|\mathbf{b}_i^{\text{rej}}\|. \quad (2.36)$$

The moment of inertia I of the whole polygon is obtained from the parallel axis theorem

$$I = \sum_{i=1}^n \left(I_i^{\Delta} + m_i^{\Delta} l_i^2 \right) \quad (2.37)$$

where l_i is the distance between the center of mass c_i^{Δ} of the i th triangle and the center of mass of the polygon C .

2.2.3 Overlap Area

For the overlap area of two contacting polygonal particles, first we need the intersection points S_1 and S_2 . While they can be computed from two nested loops over the edges of each particle, this approach is inefficient. The computation of the intersection point requires a division operation (as shown in Eq. (2.41) later in this section) which is computationally more expensive than the additions or multiplications which form the bulk of the operations. Since most pairs of edges in the nested loop will not have intersections, it is desirable to eliminate them via the following steps without actually computing the intersection points to improve the computational efficiency.

Comparison of Cartesian Coordinates

The edges of “bounding boxes” are their extremal vertices in both x - and y -directions (see Fig. 2.11). If the bounding boxes in either of the directions are not

overlapping, the pair of edges will be eliminated from the list of possible intersecting edges and we do not have to continue with the following steps and try to look for the intersection points.

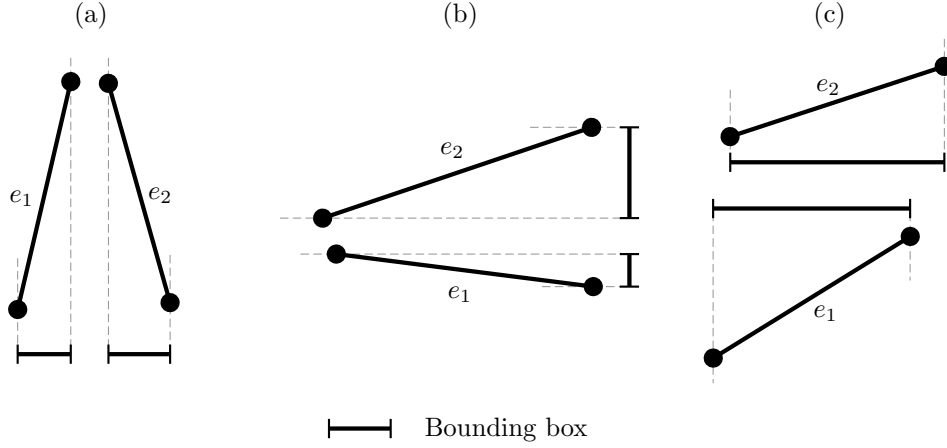


Fig. 2.11 “Bounding boxes” constructed from the endpoints of the edges. Pairs of edges with non-overlapping bounding boxes in (a) x -direction and (b) y -direction will be eliminated from the list of possible intersecting pairs. (c) will be eliminated since the bounding boxes in x -direction do not overlap.

Overlap and Orientation of Triangles

The endpoints of edges e_1 and e_2 can be used to obtain triangles whose relative orientation can be used to determine whether the edges intersect. In total we will get four triangles in total from the two edges (see Fig. 2.12). The two edges are intersecting if and only if the two triangles formed by the same edge with the endpoints of the other edge have different orientation. This means that if the two triangles from the first edge have the same orientation, the edges are not intersecting and we do not have to check the orientation of the triangles by the other edge. The orientation of the triangle can be computed from the cross product between the given edge and the vector from its endpoint to the endpoint of the other edge. This allows us to check for intersection without division. In fact, for polygons which are close, but which are just not overlapping the computational effort is largest, as all possible candidates must be tried out, while for overlapping polygons the nested loop can be terminated. Because sequences of additions and multiplications can be pipelined efficiently, the speed of determining whether there is an intersection from triangle orientation is considerably higher than from the actual intersection computation. Only the intersection point computation then needs a division.

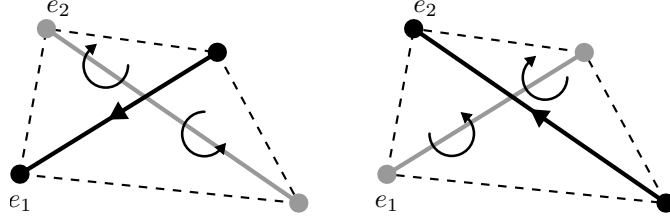


Fig. 2.12 Construction of four triangles by connecting each edge to the endpoints of the other edge. The two edges are intersecting if and only if the two triangles formed by the same edge with the endpoints of the other edge have different orientation.

Computation of the Intersection Points

The intersection point of two intersecting edges can be represented with a vector \mathbf{s} which is parallel to the one of the edge \mathbf{e}_1 as shown in Fig. 2.13. Since \mathbf{s} is parallel to \mathbf{e}_1 , the following relation holds

$$\mathbf{s} = \lambda \mathbf{e}_1 \quad (2.38)$$

where λ is a scalar which shows the length ratio of the two vectors. Area of a parallelogram A_1 can be obtained by taking the cross product between \mathbf{e}_1 and \mathbf{e}_2

$$A_1 = |\mathbf{e}_1 \times \mathbf{e}_2|. \quad (2.39)$$

On the other hand, the cross product between \mathbf{s} and \mathbf{e}_2 gives

$$A_2 = |\mathbf{s} \times \mathbf{e}_2| = |\mathbf{r} \times \mathbf{e}_2| \quad (2.40)$$

which is equivalent to the cross product between \mathbf{r} (vector which connects the tail of \mathbf{e}_2 to the tail of \mathbf{e}_1) and \mathbf{e}_2 because the area of the parallelogram will not change if it is sheared along \mathbf{e}_2 . Since length ratio between vectors \mathbf{s} and \mathbf{e}_1 is also equivalent to the ratio of the area of the parallelograms A_2 , A_1 formed by the each vector, λ can be defined as

$$\lambda = \frac{A_2}{A_1}. \quad (2.41)$$

The overlap geometry between two contacting polygonal particles is formed by the intersection points S_1 and S_2 and the vertices of the particles which penetrate into the other particle (see Fig. 2.14). The area A for the elastic contact force in Eq. (2.4) can then be computed from the resulting overlap geometry using the method discussed in Section 2.2.2.

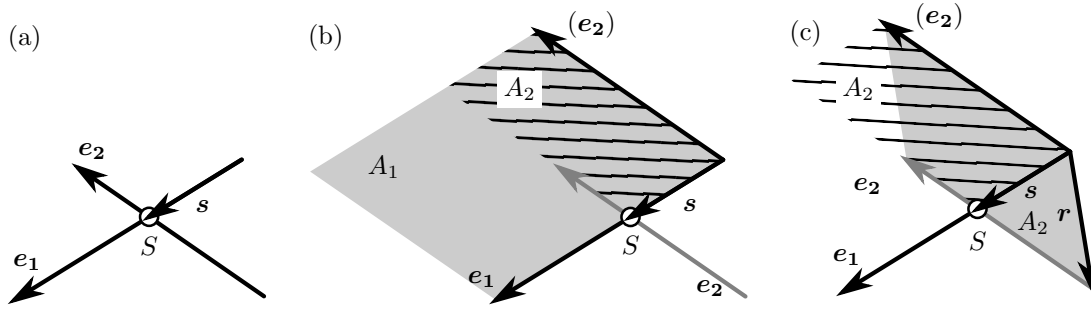


Fig. 2.13 Computation of the intersection points of the intersecting edges. (a) The intersection point is represented with vector s parallel to the edge vector e_1 . (b) λ can be computed from the ratio between area A_2 and A_1 . (c) An auxiliary vector r can be used to compute area A_2 .

2.2.4 Force Directions and Force Points

While for the contact of round particles, the force directions and force point depends only on the relative position of the centers of mass of the particles (Section 2.1), the force point and force directions for polygonal particles depends on the geometry of the contact. The force point is defined at the center of mass of the overlap geometry (see Fig. 2.15). The line segment between two intersection points S_1 and S_2 is used to define the tangential direction \hat{t} and accordingly the normal direction \hat{n} is obtained by rotation \hat{t} by 90° as shown in Fig. 2.15 (a). An alternative way of computing the force direction would be to compute the normal \hat{n} as the weighted average of the normals \hat{n}_1 and \hat{n}_2 perpendicular to the lines l_1 and l_2 connecting each intersection points to the centroid of the overlap area (see Fig. 2.15 (b))

$$\hat{n} = \frac{\hat{n}_1 l_1 + \hat{n}_2 l_2}{l_1 + l_2}. \quad (2.42)$$

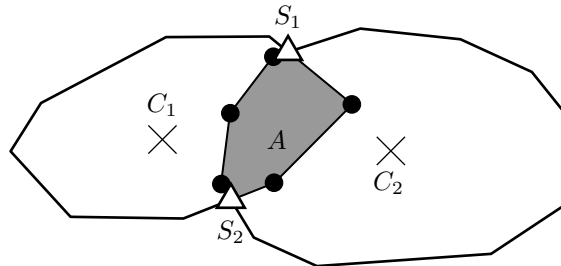


Fig. 2.14 Overlap geometry (in gray) with area A of two contacting polygonal particles constructed via the intersection points S_1 and S_2 (\triangle) and the vertices of the particles (\bullet) which penetrate into the other particle.

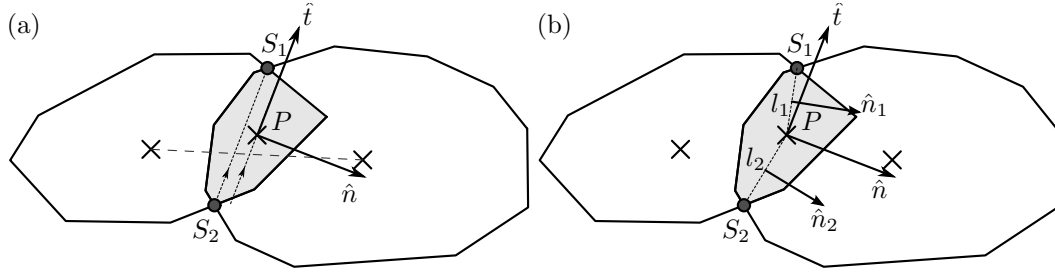


Fig. 2.15 (a) The normal \hat{n} and tangential \hat{t} directions of two contacting polygonal particles obtained directly from the “contact line” $\overline{S_1 S_2}$. (b) Normal direction \hat{n} as the weighted average of \hat{n}_1 and \hat{n}_2 . The force point P is located at the center of mass of the overlap geometry.

2.2.5 Dealing with Penetrating Contacts

The algorithms for computing the overlap geometry discussed in Section 2.2.3 assumes that there are two intersection points. This assumption works well generally except when we are dealing with particles with acute or very sharp corners. This kind of particles tends to have corners penetrating through the other particle which then leads to four intersection points as shown in Fig. 2.16. In order to obtain a consistent overlap compared to two intersection point so that the force law stays continuous for such contacts, the following procedure is applied when more than two intersection points are detected from the overlap computation.

1. Compute the rays from the center of mass C_1 of polygon 1 to each of the intersection points S_1, \dots, S_4 (see Fig. 2.16 (a)).
2. Identify the extremal rays based on the angle of the rays around the center

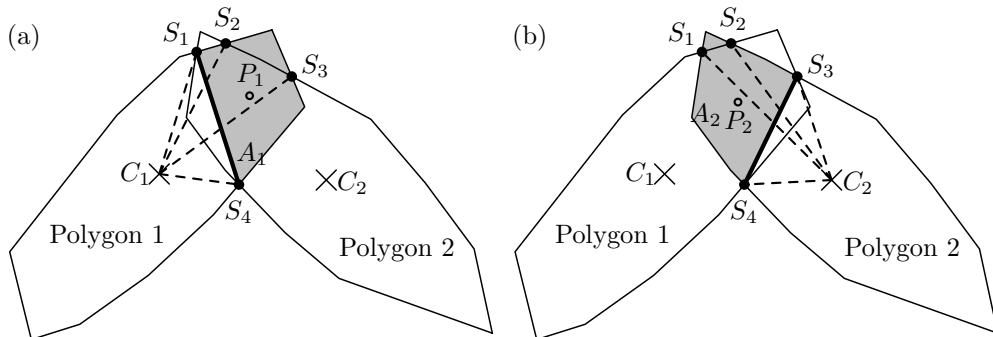


Fig. 2.16 Exaggerated sketches showing the resulting four intersection points S_1, \dots, S_4 (\bullet) from a penetrating contact. Connecting the rays (dashed lines) from the center of mass to intersection points S_1, \dots, S_4 , cut-off lines (think solid line), cut-off segments (gray area) of polygon 1 and polygon 2 are shown in (a) and (b) respectively.

of mass. Connecting two intersection points of the extremal rays gives us the cut-off line of the polygon, e.g. for polygon 1, the cut-off line is $\overline{S_1 S_4}$.

3. Compute the area A_1 and the center of mass P_1 of the segment of polygon 1 which is cut off by the line $\overline{S_1 S_4}$ (see Fig. 2.16 (a)).
4. The same processes are repeat for polygon 2 which give cut-off line $\overline{S_3 S_4}$, area A_2 and center of mass P_2 of the cut-off segment in Fig. 2.16 (b).
5. The area A for elastic contact force Eq. (2.4) is

$$A = A_1 + A_2. \quad (2.43)$$

6. The force point is given by the weighted averaged of the centroids P_1 and P_2

$$\mathbf{P} = \frac{\mathbf{P}_1 A_1 + \mathbf{P}_2 A_2}{A}. \quad (2.44)$$

7. Find the common end-point of the two cut-off lines, e.g. S_4 in Fig. 2.17. The cut-off vectors \mathbf{d}_1 and \mathbf{d}_2 can be choose as the vector from the common point to the other end-point of the cut-off line.
8. The tangential direction \hat{t} is defined from the average of \mathbf{d}_1 and \mathbf{d}_2 weighted by the areas A_1 and A_2

$$\hat{t} = \frac{\mathbf{d}_1 A_1 + \mathbf{d}_2 A_2}{A}. \quad (2.45)$$

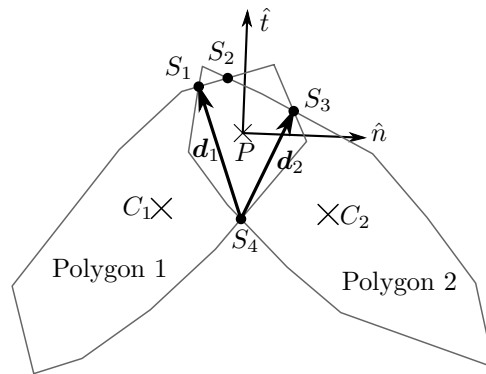


Fig. 2.17 Tangential \hat{t} and normal direction \hat{n} at force point P computed from the cut-off vectors \mathbf{d}_1 and \mathbf{d}_2 of a penetrating contact.

2.3 Time Integrator

The translational and rotational equations of motion of a particle can be written as

$$m \frac{d\ddot{\mathbf{r}}}{dt} = \mathbf{F}_{\text{net}}, \quad (2.46)$$

$$I \frac{d\ddot{\theta}}{dt} = T_{\text{net}}, \quad (2.47)$$

where \mathbf{F}_{net} is the net force on the particle which includes the total force from the contact in Eq. (2.18), the gravitational force and the force from the fluid. T_{net} is the net torque on the particle where the effect from the fluid will also be included in addition to the one from Eq. (2.19). Mass m and moment of inertia I are obtained from Eq. (2.26) and Eq. (2.37). Both equations of motion are solved using the backward differentiation formula (BDF).

For the initial value problem of the first-order differential equation

$$\dot{y}(t) = f(t, y(t)), \quad y(t_0) = y_0 \quad (2.48)$$

the general k -step backward differentiation formula is given by

$$\sum_{i=1}^k \frac{1}{i} \nabla^i y_{n+1} = \tau f(t_{n+1}, y_{n+1}) \quad (2.49)$$

where τ is the step-size and the i -th backward difference ∇^i is defined recursively as

$$\nabla^0 y_l = y_l, \quad (2.50)$$

$$\nabla^1 y_l = \nabla^{i-1} y_l - \nabla^{i-1} y_{l-1}. \quad (2.51)$$

The order of the method is given by k , e.g. we can simply expand the equations with $k = 2$ to obtain the backward differentiation formula of 2nd-order (BDF2)

$$y_{n+1} - \frac{4}{3}y_n + \frac{1}{3}y_{n-1} = \frac{2}{3}\tau f(t_{n+1}, y_{n+1}). \quad (2.52)$$

The other first six members of the family are listed in [22] and they are stable up to the 5th-order. Beyond 5th-order, BDF-methods are conditionally unstable, i.e. guaranteed to have no convergence. BDF-methods are also known as “multistep methods” (except for the first-order backward Euler) from the fact that they need additional information from more than one previous step-size (e.g. y_{n-1}) to approximate the value for the next step. The methods are also “implicit” because

the given formulations which need to be solved involve the later state of the system $f(t_{n+1}, y_{n+1})$. Such implicit schemes can be solved either by Newton iteration or by predictor–corrector approach of which we will implement for our discrete element method. For stability reason, we use the second-order BDF in our DEM simulation as BDF2 is also used as the time integrator for fluid phase. Apart from that, it has turned out that BDF2 has the best accuracy to maintain static configurations without noise: For higher order methods, vibration due to numerical noise occurs.

Gear Predictor–Corrector

The backward differentiation formula in the Gear predictor–corrector form after Gear [23] (also known as multivalue method or Nordsieck methods [24]) is used to solve the equations of motion of the particles because of its ability to neglect small oscillations in the solution and to approximate the solution of some equations with arbitrary large time-step. The solutions via the predictor-corrector form do not need a matrix inversion or a solution of a non-linear system of equations which will be explained in the following. Applying Gear predictor–corrector into the simulation involves following three separate processes

- computation of the predictor step,
- the evaluation of the forces,
- computation of the corrector step with using the change in the forces from previous step.

For a position vector \mathbf{r}_0 , we can define its successive scaled time derivatives to be

$$\mathbf{r}_1 = \tau \frac{d\mathbf{r}_0}{dt}, \mathbf{r}_2 = \frac{\tau^2}{2!} \frac{d^2\mathbf{r}_0}{dt^2}, \mathbf{r}_3 = \frac{\tau^3}{3!} \frac{d^3\mathbf{r}_0}{dt^3}, \dots, \mathbf{r}_n = \frac{\tau^n}{n!} \frac{d^n\mathbf{r}_0}{dt^n}. \quad (2.53)$$

The value of the solution for the next time-step $\mathbf{r}_0(t + \tau)$ can be approximated by using via the Taylor series at $\mathbf{r}_0(t)$ as

$$\mathbf{r}_0(t + \tau) = \mathbf{r}_0(t) + \tau \frac{d\mathbf{r}_0(t)}{dt} + \frac{\tau^2}{2} \frac{d^2\mathbf{r}_0(t)}{dt^2} + \frac{\tau^3}{6} \frac{d^3\mathbf{r}_0(t)}{dt^3} + \dots + \frac{\tau^n}{n!} \frac{d^n\mathbf{r}_0(t)}{dt^n}, \quad (2.54)$$

substituting the successive scaled time derivatives into the equation gives us

$$\mathbf{r}_0(t + \tau) = \mathbf{r}_0(t) + \mathbf{r}_1(t) + \mathbf{r}_2(t) + \mathbf{r}_3(t) + \dots + \mathbf{r}_n(t) \quad (2.55)$$

Applying the same procedures to the time derivatives $\frac{d\mathbf{r}_0(t+\tau)}{dt}$, $\frac{d^2\mathbf{r}_0(t+\tau)}{dt^2}$, \dots , $\frac{d^n\mathbf{r}_0(t+\tau)}{dt^n}$ the predictor of a six-value method can be computed as

$$\begin{pmatrix} \mathbf{r}_0^p(t+\tau) \\ \mathbf{r}_1^p(t+\tau) \\ \mathbf{r}_2^p(t+\tau) \\ \mathbf{r}_3^p(t+\tau) \\ \mathbf{r}_4^p(t+\tau) \\ \mathbf{r}_5^p(t+\tau) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & 3 & 6 & 10 \\ 0 & 0 & 0 & 1 & 4 & 10 \\ 0 & 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{r}_0(t) \\ \mathbf{r}_1(t) \\ \mathbf{r}_2(t) \\ \mathbf{r}_3(t) \\ \mathbf{r}_4(t) \\ \mathbf{r}_5(t) \end{pmatrix} \quad (2.56)$$

which is simply a polynomial extrapolation of the solution and its derivatives (not solving the differential equation yet). The matrix is the Pascal triangle matrix and the vector on the left-hand side is known as the Nordsieck vector.

The actual approximation is done by adding a correction to the predicted values \mathbf{r}_n^p , i.e. the corrector step

$$\begin{pmatrix} \mathbf{r}_0^c(t+\tau) \\ \mathbf{r}_1^c(t+\tau) \\ \mathbf{r}_2^c(t+\tau) \\ \mathbf{r}_3^c(t+\tau) \\ \mathbf{r}_4^c(t+\tau) \\ \mathbf{r}_5^c(t+\tau) \end{pmatrix} = \begin{pmatrix} \mathbf{r}_0^p(t+\tau) \\ \mathbf{r}_1^p(t+\tau) \\ \mathbf{r}_2^p(t+\tau) \\ \mathbf{r}_3^p(t+\tau) \\ \mathbf{r}_4^p(t+\tau) \\ \mathbf{r}_5^p(t+\tau) \end{pmatrix} + \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{pmatrix} \Delta\mathbf{r}. \quad (2.57)$$

This predictor–corrector method is available for both first- and second-order ordinary differential equations (ODE) which means that our equations of motions do not have to be transformed into a first-order system. For the second-order ODEs, $\Delta\mathbf{r}$ is the difference between the predicted and corrected value of the second derivatives i.e. the predicted force and the corrected force (obtained from the force computations in Section 2.1 with the predicted positions $\mathbf{r}_0^p(t+\tau)$ and velocities $\mathbf{r}_1^p(t+\tau)$)

$$\Delta\mathbf{r} = \mathbf{r}_2^c(t+\tau) - \mathbf{r}_2^p(t+\tau). \quad (2.58)$$

The coefficients c_0, c_1, \dots, c_5 for the second-order ODEs of the form $\ddot{\mathbf{r}} = f(\mathbf{r}, \dot{\mathbf{r}})$ in which the first derivatives also appear in the right-hand side (as in our case) is given in Table 2.1. The coefficients for the first-order ODEs and second-order ODEs of $\ddot{\mathbf{r}} = f(\mathbf{r})$ can be found in [25, Table E.1 and E.2].

The time-scaled derivatives $\mathbf{r}_1, \mathbf{r}_2, \dots$ were originally introduced by Nordsieck [26] for the purpose of changing the step-size τ easily. We can simply multiply the

Table 2.1 Gear corrector coefficients for second-order differential equations of the form $\ddot{\mathbf{r}} = f(\mathbf{r}, \dot{\mathbf{r}})$.

Order	Values	c_0	c_1	c_2	c_3	c_4	c_5
2	3	0	1	1			
3	4	$1/6$	$5/6$	1	$1/3$		
4	5	$19/90$	$3/4$	1	$1/2$	$1/12$	
5	6	$3/16$	$251/360$	1	$11/18$	$1/6$	$1/60$

components of the Nordsieck vector by the appropriate powers of τ'/τ to change the step-size from τ to τ' . This will be become very handy as we can combine the fluid code where the adaptive step-size is also implemented with the discrete element method with less effort.

As the definition of the normal force ($F_{c,\perp} - F_{d,\perp}$) is modeled as a harmonic oscillator, the characteristic oscillation frequency ω_c (which would correspond to a particle which vibrates on a ground with the same Young's modulus under the influence of its own weight) can be determined from the smallest particle mass m_{\min} in the system and the two-dimensional Young's modulus Y as

$$\omega_c = \sqrt{\frac{Y}{m_{\min}}} \quad (2.59)$$

where the duration of a full period of an oscillation is

$$\begin{aligned} T_c &= \frac{2\pi}{\omega_c}, \\ &= 2\pi \sqrt{\frac{m_{\min}}{Y}}. \end{aligned} \quad (2.60)$$

For a collision of particles, the duration of a collision τ_c would be about half of the period

$$\tau_c \approx \frac{1}{2}T_c \approx \pi \sqrt{\frac{m_{\min}}{Y}}. \quad (2.61)$$

Since the resolution with 10 step-size is usually enough to resolve the collision of the particles with the BDF solver, the step-size of a dry discrete element simulation can be set at

$$\tau = \frac{1}{10}\pi \sqrt{\frac{m_{\min}}{Y}}. \quad (2.62)$$

For smaller time-steps, there is the danger that instabilities develop, aggregates of particles “explode.” The time-step is valid only for dry granular materials: For particles in fluid, the minimal time-step of ten times larger than the dry case may

be used, as the fluid damps the relative motion of the particles so that the contact time is longer than for dry particles.

Chapter 3

Simulation of the Fluid Part

3.1 Governing Equations

Compared to continuum materials, for granular materials from the same material, the sound velocity is reduced considerably. It has been found that sound velocities of an uncompressed granular packing are less than 10% for two-dimensional granular assemblies (rods) compared to the space-filling packing (homogeneous material) in simulations, and less than 1% for three-dimensional assemblies (plastic-beads) in experiments [5], as the transfer of momentum can take place via the relatively narrow particle interstices. As the sound velocity of the granular assembly will be much smaller than that of the fluid, we will treat the flow as incompressible. In that case, the two-dimensional Navier–Stokes equations for incompressible Newtonian fluid can be written as

$$\begin{aligned}\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} &= -\frac{1}{\rho} \frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + X, \\ \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} &= -\frac{1}{\rho} \frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + Y,\end{aligned}\tag{3.1}$$

and the continuity equation as

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0,\tag{3.2}$$

with velocities in x - and y -directions u , v and pressure p as the unknowns. The kinematic viscosity ν in the second terms of right-hand sides of Eqs. (3.1) can be derived from the dynamic viscosity μ_f [Pa · s] and the density ρ [kg/m³] of the fluid

as¹

$$\nu = \frac{\mu_f}{\rho} \quad (3.3)$$

This means that the effect from the viscosity on the motion of the fluid is not expressed by the dynamic viscosity μ_f alone but the density of the fluid must also be considered. With $\mathbf{u} = [u \ v]^\top$ and $\mathbf{f} = [X \ Y]^\top$, the Navier–Stokes and continuity equations can also be rewritten as

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \quad (3.4)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (3.5)$$

with the nabla operator in Cartesian coordinates $\nabla = \left[\frac{\partial}{\partial x} \ \frac{\partial}{\partial y} \right]^\top$. Multiplying the constant ρ into the Eq. (3.4) give us

$$\frac{\partial (\rho \mathbf{u})}{\partial t} = - \underbrace{(\mathbf{u} \cdot \nabla) (\rho \mathbf{u})}_{\text{Advection term}} + \underbrace{\mu_f \nabla^2 (\rho \mathbf{u})}_{\text{Diffusion term}} - \nabla p + \rho \mathbf{f}, \quad (3.6)$$

where $\rho \mathbf{u}$ is the momentum per unit volume of the fluid. The above equation means that the rate of change of the momentum $\frac{\partial (\rho \mathbf{u})}{\partial t}$ is due to:

Advection term: The first term on the right-hand side of Eq. (3.6). This non-linear term means that the momentum is transported by the flow itself.

Diffusion term: The second term on the right-hand side of Eq. (3.6) describes the momentum is transferred from regions of high to regions of low momentum.

Pressure and external forces: The third and forth terms on the right-hand side of Eq. (3.6) represent the pressure and body forces acting on the fluid.

3.2 Finite Element Methods

Solving the Navier–Stokes equations directly via numerical methods requires the computation of second derivatives. To weaken the requirement on the smoothness of the solution functions, several approximation methods have been developed. First the residual function is defined as the difference between the exact solution of the governing equations and the numerical approximation as linear combinations of some trial functions. Then the residual function is set to zero on average by requiring it

¹Since we have used μ to represent the friction coefficient in Section 2.1.3 for the simulation of the granular part, μ_f [Pa · s] will be used to represent the dynamic viscosity throughout the thesis.

to be orthogonal to the vector space spanned by a given arbitrary test function. This approach is known as the method of mean weighted residuals while the test function is sometime referred as the weighting function. The approach where the basis functions of the same family are used for both approximated solutions and the test functions is known as the Galerkin method² while in the Petrov–Galerkin method, different basis functions are used for the approximated solutions and the test functions. The basis functions can be global polynomials, piecewise polynomials, trigonometric polynomials, etc. The finite element method is a Galerkin method with piecewise polynomials as its basis functions while Galerkin methods with trigonometric functions are known as spectral methods. The order of the necessary derivatives is then reduced via introduction integration by parts: This leads to the weak form of the problem. This allows us to operate with flow-fields where only the velocities have to be smooth functions, not their time derivatives.

3.2.1 The Weak Form of the Navier–Stokes Equations

In this section, we show the formal derivation of the Navier–Stokes equations which we will use for the simulation. For simplicity, we rewrite the Navier–Stokes equations Eq. (3.1) with the normalized pressure $P = p/\rho$ as

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = -\frac{\partial P}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + X, \quad (3.7)$$

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} = -\frac{\partial P}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + Y. \quad (3.8)$$

For the x -component of the Navier–Stokes equations Eq. (3.7), we start with multiplying it by a test function $\phi^{(x)}$ and integrate the equation over the fluid region

$$\iint \phi^{(x)} \underbrace{\left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + \frac{\partial P}{\partial x} - \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - X \right)}_{\text{Residual function, } R} dx dy = 0 \quad (3.9)$$

where the residual function R of Eq. (3.7) will be zero on average by requiring it to be orthogonal to vector space spanned by the given test function $\phi^{(x)}$, i.e. the inner product of the two functions is zero

$$\iint \phi^{(x)} R dx dy = 0 \quad (3.10)$$

²Galerkin ascribed the method to Ritz, whose variational principle underlies the whole approach.

This approach is known as the method of mean weighted residuals and the test function $\phi^{(x)}$ is sometime referred to as the weighting function.

Next, we will lower the order of Eq. (3.9) by applying the integration by parts using the Gaussian divergence theorem. For $\mathbf{v} = [v_1 \ v_2]^\top$ where v_1 and v_2 are two continuously differentiable functions on the closure of domain Ω , the two-dimensional Gaussian divergence theorem is given as

$$\begin{aligned} \iint_{\Omega} \nabla \cdot \mathbf{v} \, d\Omega &= \int_{\Gamma} \mathbf{v} \cdot \hat{n} \, d\Gamma \\ \iint \left(\frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y} \right) dx \, dy &= \int_{\Gamma} (v_1 n_x + v_2 n_y) \, d\Gamma \end{aligned} \quad (3.11)$$

where Γ is the boundary around the domain, and $\hat{n} = [n_x \ n_y]^\top$ is the unit normal vector to Γ pointing outward. Substituting

$$v_1 = \phi^{(x)} P, \quad v_2 = 0 \quad (3.12)$$

into Eq. (3.11) gives us

$$\iint \frac{\partial}{\partial x} (\phi^{(x)} P) \, dx \, dy = \int_{\Gamma} \phi^{(x)} P n_x \, d\Gamma. \quad (3.13)$$

Expanding the partial derivative on the left-hand side with the product rule gives

$$\iint \phi^{(x)} \frac{\partial P}{\partial x} \, dx \, dy = - \iint P \frac{\partial \phi^{(x)}}{\partial x} \, dx \, dy + \int_{\Gamma} \phi^{(x)} P n_x \, d\Gamma \quad (3.14)$$

which corresponds to the fourth (pressure) term in Eq. (3.9). Similar procedures can be performed on the diffusion term by substituting

$$v_1 = \phi^{(x)} \frac{\partial u}{\partial x}, \quad v_2 = \phi^{(x)} \frac{\partial u}{\partial y} \quad (3.15)$$

into Eq. (3.11) to obtain

$$\begin{aligned} &\iint \phi^{(x)} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) dx \, dy \\ &= \int_{\Gamma} \phi^{(x)} \left(\frac{\partial u}{\partial x} n_x + \frac{\partial u}{\partial y} n_y \right) d\Gamma - \iint \frac{\partial u}{\partial x} \frac{\partial \phi^{(x)}}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial \phi^{(x)}}{\partial y} dx \, dy. \end{aligned} \quad (3.16)$$

Substituting Eq. (3.14) and Eq. (3.16) into Eq. (3.9) then with some arrangements,

the weak form of the x -component of the Navier–Stokes equations can be written as

$$\begin{aligned} & \iint \phi^{(x)} \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} - X \right) - P \frac{\partial \phi^{(x)}}{\partial x} + \nu \left(\frac{\partial u}{\partial x} \frac{\partial \phi^{(x)}}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial \phi^{(x)}}{\partial y} \right) dx dy \\ &= \int_{\Gamma} \phi^{(x)} \left\{ n_x \left(\nu \frac{\partial u}{\partial x} - P \right) + n_y \left(\nu \frac{\partial u}{\partial y} \right) \right\} d\Gamma, \end{aligned} \quad (3.17)$$

which also allows us to lower the order of differentiation in the diffusion term of the equation. With the same procedures, the equivalent result for the y -component of the Navier–Stokes equations is

$$\begin{aligned} & \iint \phi^{(y)} \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} - Y \right) - P \frac{\partial \phi^{(y)}}{\partial y} + \nu \left(\frac{\partial v}{\partial x} \frac{\partial \phi^{(y)}}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial \phi^{(y)}}{\partial y} \right) dx dy \\ &= \int_{\Gamma} \phi^{(y)} \left\{ n_x \left(\nu \frac{\partial v}{\partial x} \right) + n_y \left(\nu \frac{\partial v}{\partial y} - P \right) \right\} d\Gamma, \end{aligned} \quad (3.18)$$

where $\phi^{(y)}$ is the test function for the y -component. The weak form of the continuity equation Eq. (3.2) is simply

$$\iint \psi \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) dx dy = 0, \quad (3.19)$$

where ψ is the test function related to the pressure.

Γ^D are the Dirichlet-type (fixed) boundaries where the velocities and pressures are given, and Γ^N are the Neumann-type boundaries where the normal derivative of the function is specified. The boundary conditions that are appropriate for the above weak formulations are:

For u :

$$u = U \quad \text{on} \quad \Gamma_u^D, \quad (3.20)$$

$$n_x \left(\nu \frac{\partial u}{\partial x} - P \right) + n_y \left(\nu \frac{\partial u}{\partial y} \right) = F_x \quad \text{on} \quad \Gamma_u^N. \quad (3.21)$$

For v :

$$v = V \quad \text{on} \quad \Gamma_v^D, \quad (3.22)$$

$$n_x \left(\nu \frac{\partial v}{\partial x} \right) + n_y \left(\nu \frac{\partial v}{\partial y} - P \right) = F_y \quad \text{on} \quad \Gamma_v^N, \quad (3.23)$$

where U , V , F_x , F_y and $\Gamma_u^D + \Gamma_u^N = \Gamma_v^D + \Gamma_v^N = \Gamma$ are specified. By restricting the

test functions to vanish on the Dirichlet portions of the boundary ($\phi^{(x)} = 0$ on Γ_u^D , $\phi^{(y)} = 0$ on Γ_v^D) the weak form of the Navier–Stokes equations under the above boundary conditions are

$$\begin{aligned} & \iint \phi^{(x)} \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} - X \right) - P \frac{\partial \phi^{(x)}}{\partial x} + \nu \left(\frac{\partial u}{\partial x} \frac{\partial \phi^{(x)}}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial \phi^{(x)}}{\partial y} \right) dx dy \\ &= \int_{\Gamma_u^N} \phi^{(x)} F_x d\Gamma, \end{aligned} \quad (3.24)$$

$$\begin{aligned} & \iint \phi^{(y)} \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} - Y \right) - P \frac{\partial \phi^{(y)}}{\partial y} + \nu \left(\frac{\partial v}{\partial x} \frac{\partial \phi^{(y)}}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial \phi^{(y)}}{\partial y} \right) dx dy \\ &= \int_{\Gamma_v^N} \phi^{(y)} F_y d\Gamma. \end{aligned} \quad (3.25)$$

3.2.2 Discretization of the Weak Form

We start by approximating the velocities and pressure as the linear combination of the basis functions $\varphi_i^{(x)}$, $\varphi_i^{(y)}$, and ψ_i

$$u = \sum_{i=1}^{N_x} u_i \varphi_i^{(x)} = \boldsymbol{\varphi}_{(x)}^T \mathbf{u}, \quad (3.26)$$

$$v = \sum_{i=1}^{N_y} v_i \varphi_i^{(y)} = \boldsymbol{\varphi}_{(y)}^T \mathbf{v}, \quad (3.27)$$

$$P = \sum_{i=1}^{N_P} P_i \psi_i = \boldsymbol{\psi}^T \mathbf{P}, \quad (3.28)$$

where u_i , v_i , and P_i are the nodal values of the velocities and pressures which are to be determined as the solutions of the Navier–Stokes equations. N_x and N_y indicate the number of velocity nodes in the fluid domain Ω and on the boundaries. The total number of velocity nodes N_T is

$$N_T = N_x + N_y. \quad (3.29)$$

N_P is the number of pressure nodes.

The test functions which we implemented in the weak form of the Navier–Stokes equations can also be written as the linear combinations of the same basis functions $\varphi_i^{(x)}$, $\varphi_i^{(y)}$, and ψ_i

$$\phi^{(x)} = \sum_{i=1}^{N_x} \varphi_i^{(x)} \Phi_i^{(x)*} = \boldsymbol{\varphi}_{(x)}^T \boldsymbol{\Phi}_{(x)}^*, \quad (3.30)$$

$$\phi^{(y)} = \sum_{i=1}^{N_y} \varphi_i^{(y)} \Phi_i^{(y)*} = \boldsymbol{\varphi}_{(y)}^\top \boldsymbol{\Phi}_{(y)}^*, \quad (3.31)$$

$$\psi = \sum_{i=1}^{N_P} \psi_i \Psi_i^* = \boldsymbol{\psi}^\top \boldsymbol{\Psi}^*, \quad (3.32)$$

where $\Phi_i^{(x)*}$, $\Phi_i^{(y)*}$ and Ψ_i^* are the nodal values of the test functions. By substituting (approximated) velocities u , v , pressures P , and the test functions $\phi^{(x)}$, $\phi^{(y)}$ and ψ into Eq. (3.24), Eq. (3.25) and Eq. (3.19) with the corresponding definitions above, the weak form of the Navier–Stokes equations is obtained in matrix-vector representation

$$M\dot{\mathbf{u}} + [K + N(\mathbf{u})] \mathbf{u} + C\mathbf{P} = \mathbf{f}, \quad (3.33)$$

$$C^\top \mathbf{u} = \mathbf{0}, \quad (3.34)$$

In our approach, where the particles are described in Cartesian coordinates, we will also use the discretized Navier–Stokes equations in Cartesian coordinates, so that, e.g. \mathbf{u}_1 denotes the N_x -column-vector of nodal velocities in the x -direction. The partitioned matrices in block-notation are named as

$$\begin{aligned} \text{Mass matrix, } M &= \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}, \\ \text{Viscous or diffusion matrix, } K &= \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix}, \\ \text{Non-linear advection matrix, } N(\mathbf{u}) &= \begin{bmatrix} N_1(\mathbf{u}) & 0 \\ 0 & N_2(\mathbf{u}) \end{bmatrix}, \\ \text{Constraint matrix, } C &= \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}, \quad C^\top = \begin{bmatrix} C_1^\top & C_2^\top \end{bmatrix}, \\ \mathbf{f} &= \left\{ \mathbf{f}_1^\top \quad \mathbf{f}_2^\top \right\}^\top, \\ \mathbf{u} &= \left\{ \mathbf{u}_1^\top \quad \mathbf{u}_2^\top \right\}^\top. \end{aligned}$$

In the following, when we write the matrices with the respective basis functions, indices α and β stand for the spatial dimensions. We further use the Einstein summation convention: where for any subscripted variable which appears twice (and only twice) in any term of an expression, the subscripted variables are assumed to be summed over. However, when the spatial index appears in parentheses as (α) , no summation is implied. The matrices and their dimensions in terms of basis functions

Eq. (3.30)–(3.32) are then

$$M_\alpha = \int \boldsymbol{\varphi}_{(\alpha)} \boldsymbol{\varphi}_{(\alpha)}^\top : \quad (N_\alpha \times N_\alpha), \quad (3.35)$$

$$N_\alpha(\mathbf{u}) = \int \boldsymbol{\varphi}_{(\alpha)} \left(\boldsymbol{\varphi}_{(1)}^\top \mathbf{u}_1 \frac{\partial \boldsymbol{\varphi}_{(\alpha)}^\top}{\partial x_1} + \boldsymbol{\varphi}_{(2)}^\top \mathbf{u}_2 \frac{\partial \boldsymbol{\varphi}_{(\alpha)}^\top}{\partial x_2} \right) : \quad (N_\alpha \times N_\alpha), \quad (3.36)$$

$$K_\alpha = \int \nu \left(\frac{\partial \boldsymbol{\varphi}_{(\alpha)}}{\partial x_\beta} \frac{\partial \boldsymbol{\varphi}_{(\alpha)}^\top}{\partial x_\beta} \right) : \quad (N_\alpha \times N_\alpha), \quad (3.37)$$

$$C_\alpha = \int -\frac{\partial \boldsymbol{\varphi}_{(\alpha)}}{\partial x_\alpha} \boldsymbol{\psi}^\top : \quad (N_\alpha \times N_P). \quad (3.38)$$

On the right-hand side of the discretized Navier–Stokes equation, we have the volume forces

$$\mathbf{f}_\alpha = \int \boldsymbol{\varphi}_{(\alpha)} X_\alpha + \int_{\Gamma_\alpha^N} \boldsymbol{\varphi}_{(\alpha)} F_\alpha, \quad (3.39)$$

which is a $N_\alpha \times 1$ column vector. Detailed derivations are given in Appendix A.

In mechanics, constrained problems can be formulated as differential-algebraic equations (DAE) which allow a favorable formulation (commonly used in numerical analysis [17]) as

$$M(t, \mathbf{q}) \dot{\mathbf{u}} + G^\top(t, \mathbf{q}) \boldsymbol{\lambda} = \mathbf{f}(t, \mathbf{u}, \mathbf{v}), \quad (3.40)$$

$$G(t, \mathbf{q}) \mathbf{u} = \mathbf{0}. \quad (3.41)$$

This is an equation of motion for the masses M , the Jacobian of the constraints G , the time derivatives of the velocities $\dot{\mathbf{u}}$ and the external forces \mathbf{f} , with the Lagrange multipliers $\boldsymbol{\lambda}$ (see Appendix B for more detailed derivations). The continuity equation in our fluid problem Eq. (3.2) can be understood as a constraint on the compressibility and the spatial discretization of the Navier–Stokes equations. With the Galerkin method it turns out to be an index-2 formulation of the differential-algebraic equations as Eq. (3.40)–(3.41) [22]. Comparison with Eq. (3.40) shows that the pressures in Eq. (3.33) indeed play the role of Lagrange multipliers. They inhibit in- and outflows which would violate the incompressibility condition defined by the continuity equation. Non-smooth changes of the pressures from one time-step to the next are in principle possible in this formulation without necessarily affecting later time-steps: There is no need for a smooth time-evolution of pressures as in the pressure-relaxation approach of the Marker-and-Cell [27] approach, which may lead to a hysteretic dynamics which will be difficult to control for moving boundaries. On the other hand, if the pressures fluctuate too strongly, the velocity field may become

noisy, which may destabilize the simulations. Too large fluctuations may indicate that the time-step of the simulation is too large (compared to the dynamics of the fluid). This may not necessarily destabilize the simulation, at least when implicit time integrators are used, as in our case.

3.2.3 Choice of Elements

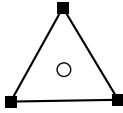
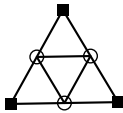
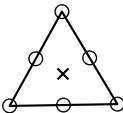
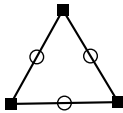
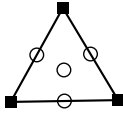
The main goal of this study is to simulate flows of fluids in which particles with arbitrary polygonal shapes are suspended (respectively, for high particle density, the flow through the pore space of a granular assembly). Therefore, from the geometrical point of view, we are only considering triangular elements because only they can discretize the pore space between polygonal particles adequately by triangulation. For general flow simulations, it has been argued that quadrilateral meshes will give higher accuracy than the triangular ones, but this has only been shown for stratified flow fields where the element boundaries are more or less parallel to the stream lines. For the case of particle-laden flows which generate small vortices, one can expect that triangular meshes give a better discretization than rectangular elements due to symmetry reasons: Circular shapes (of vortices) can be more easily be composed from (triangular) sectors than square shaped elements.

In general, in order to obtain a stable simulation of incompressible flows via the finite element method, the approximations of the pressures should be at least one order smaller than of the velocities. This is known as Ladyzhenskaya–Babuska–Brezzi (LBB) condition [28] and the methods which use different approximation order for velocities and pressures are known as mixed interpolation method. A few choices of triangular elements which are LBB stable are shown in Table 3.1 where “P” stands for polynomial and the subscripts indicate the order of the polynomial.

3.2.4 Taylor–Hood Element

For the sake of simplicity, geometrical advantage and numerical stability we choose the P_2P_1 -elements from Table 3.1 where the pressures are approximated with affine functions while the velocities are approximated with quadratic functions for our spatial discretization. Since the Navier–Stokes equations are of higher order in the velocities than in the pressures, and this should be represented in the choice of elements, we avoid the usage of elements with linear velocity: $P_1^+P_1$ and P_1P_1 on a 4-patch. P_2P_0 is not suitable because no pressures on the boundaries are available: The forces on the particles from the fluid cannot be computed. We do not use $P_2^+P_1$

Table 3.1 List of triangular elements which are LBB stable.

Name	Sketch	Comments
$P_1^+P_1$		Simple. First-order. Cubic bubble function on \circ . Lower memory usage. Computationally cheaper. Might be unstable due to Checkerboard pressure mode.
P_1P_1 on a 4-patch		First-order. Best element with linear velocity. Element is partitioned into four smaller triangles. First-order approximation of velocities in each triangle. Also known as Bercovier–Pironneau element [29].
P_2P_0		Simple. Discontinuous pressure. High memory usage and CPU-time consumption. Not much improvement in accuracy (first-order).
P_2P_1		Simplest second-order element. Also known as Taylor–Hood element [30].
$P_2^+P_1$		Second-order. Cubic bubble function. “Better accuracy” than P_2P_1 . Higher memory usage and CPU-time consumption.

- \circ Velocity
 \blacksquare Velocity and continuous pressure
 \times Discontinuous pressure

elements, because the (rather non-local) cubic bubble function has no immediate physical meaning, and we don’t know how it will affect the flow over a porous space. The affine approximation (first-order polynomial) of a test function $\psi(x, y)$ inside an element e can be written as

$$\psi(x, y) = a_e + b_e x + c_e y = \begin{Bmatrix} 1 & x & y \end{Bmatrix} \begin{Bmatrix} a_e \\ b_e \\ c_e \end{Bmatrix}, \quad (3.42)$$

where a_e, b_e, c_e are the unknown coefficients for the element e . From Eq. (3.42), the nodal values Ψ_1, \dots, Ψ_3 of the test function on the vertices $P_1(x_1, y_1), P_2(x_2, y_2),$

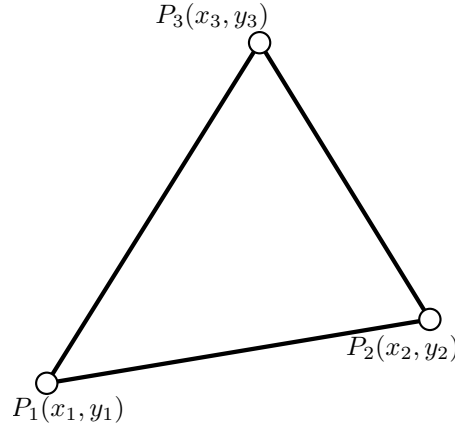


Fig. 3.1 Triangular element with vertices $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ and $P_3(x_3, y_3)$ in counterclockwise orientation.

$P_3(x_3, y_3)$ of the triangle in Fig. 3.1 can be written as

$$\begin{Bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{Bmatrix} a_e \\ b_e \\ c_e \end{Bmatrix}.$$

Multiplying the inverse of the matrix on both sides of the equation to solve for the unknown coefficients we obtain,

$$\begin{Bmatrix} a_e \\ b_e \\ c_e \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}^{-1} \begin{Bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \end{Bmatrix}, \quad (3.43)$$

$$\begin{Bmatrix} a_e \\ b_e \\ c_e \end{Bmatrix} = \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & B_2 & B_3 \\ C_1 & C_2 & C_3 \end{bmatrix} \begin{Bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \end{Bmatrix}, \quad (3.44)$$

where the elements of the inverse matrix are,

$$A_1 = \frac{x_2 y_3 - x_3 y_2}{2\Delta_e}, \quad B_1 = \frac{y_2 - y_3}{2\Delta_e}, \quad C_1 = \frac{x_3 - x_2}{2\Delta_e}, \quad (3.45)$$

$$A_2 = \frac{x_3 y_1 - x_1 y_3}{2\Delta_e}, \quad B_2 = \frac{y_3 - y_1}{2\Delta_e}, \quad C_2 = \frac{x_1 - x_3}{2\Delta_e}, \quad (3.46)$$

$$A_3 = \frac{x_1 y_2 - x_2 y_1}{2\Delta_e}, \quad B_3 = \frac{y_1 - y_2}{2\Delta_e}, \quad C_3 = \frac{x_2 - x_1}{2\Delta_e}, \quad (3.47)$$

and Δ_e is the area of the triangular element

$$\Delta_e = \frac{1}{2} \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix}. \quad (3.48)$$

The test function can be written as the linear combinations of the shape functions ψ by substituting Eq. (3.44) into Eq. (3.42)

$$\begin{aligned} \psi(x, y) &= \begin{Bmatrix} 1 & x & y \end{Bmatrix} \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & B_2 & B_3 \\ C_1 & C_2 & C_3 \end{bmatrix} \begin{Bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \end{Bmatrix}, \\ &= \begin{Bmatrix} \psi_1(x, y) & \psi_2(x, y) & \psi_3(x, y) \end{Bmatrix} \begin{Bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \end{Bmatrix}, \\ &= \boldsymbol{\psi}^T \boldsymbol{\Psi}, \end{aligned} \quad (3.49)$$

where the shape functions are defined as

$$\begin{aligned} \psi_1(x, y) &= A_1 + B_1x + C_1y, \\ \psi_2(x, y) &= A_2 + B_2x + C_2y, \\ \psi_3(x, y) &= A_3 + B_3x + C_3y. \end{aligned} \quad (3.50)$$

Next, we want to consider another coordinate system which will simplify the shape functions considerably especially when it comes to integration of the functions. Let $P(x, y)$ be a arbitrary position inside a triangle with area Δ_e as shown in Fig. 3.2.

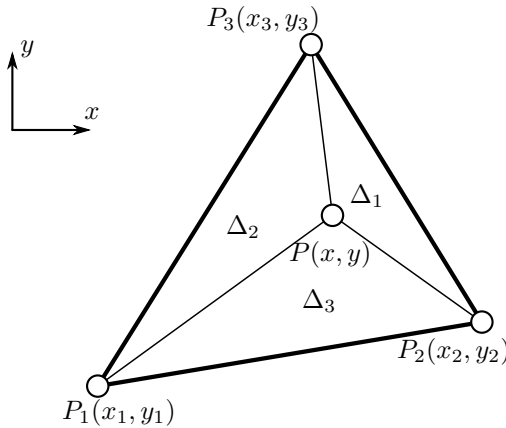


Fig. 3.2 A triangle with area Δ_e partitioned into three smaller triangles, where Δ_1 is the area of $\triangle PP_2P_3$, Δ_2 is the area of $\triangle PP_3P_1$, and Δ_3 is the area of $\triangle PP_1P_2$.

The triangle can be partitioned into three smaller triangles with areas Δ_1 , Δ_2 , Δ_3 by connecting P with the vertices P_1 , P_2 , P_3 . The ratios between the areas of the partitioned triangles and the whole total area Δ_e can be written as

$$L_1 = \frac{\Delta_1}{\Delta_e}, \quad L_2 = \frac{\Delta_2}{\Delta_e}, \quad L_3 = \frac{\Delta_3}{\Delta_e}, \quad (3.51)$$

where L_1 , L_2 , L_3 vary between 0 and 1. These ratios can be used as the coordinates of the point inside the triangle $P(L_1, L_2, L_3)$ (see Fig. 3.3) which is known as local area coordinates, oblique coordinates [31] or barycentric coordinates. This means we work with a redundant coordinate system, not with the minimal amount of base vectors as is usually common practice. Because there are three variables (L_1, L_2, L_3) in this coordinate system instead of two, we have a constraint between these variables,

$$L_1 + L_2 + L_3 = 1, \quad (3.52)$$

where the sum of the ratios is always equal to one. This leaves us with two degrees of freedom as in a two-dimensional Cartesian coordinate system. Note that when point P lies on one of the edges, the area coordinate with the same index as the opposite vertex will be zero. The centroid of the triangle in the area coordinate is located at $(1/3, 1/3, 1/3)$. The local area coordinate is useful for checking if a given point is inside a triangle: If the range of any of the coordinate L_i is not within 0 and 1, the given point is not located inside the triangle.

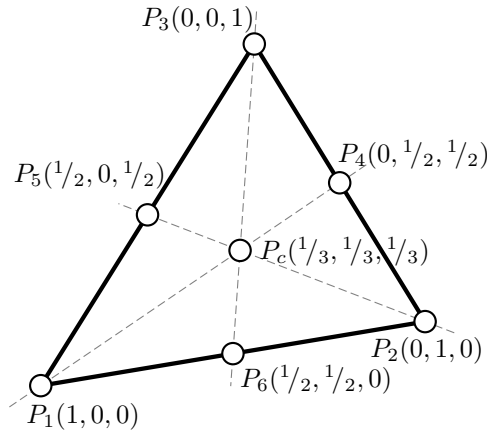


Fig. 3.3 Vertices and midpoints of a triangle in local area coordinates.

Using the definitions in Eq. (3.45), the area of the partitioned triangle $\triangle PP_2P_3$

can be written as

$$\begin{aligned}\Delta_1 &= \frac{1}{2} \begin{vmatrix} 1 & x & y \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix}, \\ &= \frac{1}{2} \{ (x_2 y_3 - x_3 y_2) + (y_2 - y_3)x + (x_3 - x_2)y \}, \\ &= \Delta_e \{ A_1 + B_1 x + C_1 y \}.\end{aligned}$$

From the above derivations, it turns out that the local coordinates are also an affine approximation with A_i , B_i , C_i as the coefficients

$$\begin{Bmatrix} L_1 \\ L_2 \\ L_3 \end{Bmatrix} = \begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{bmatrix} \begin{Bmatrix} 1 \\ x \\ y \end{Bmatrix} = \begin{Bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{Bmatrix}, \quad (3.53)$$

are shape functions in Eq. (3.50). Using Eq. (3.43) and Eq. (3.44), the transformation from area coordinates to Cartesian coordinates is

$$\begin{Bmatrix} 1 \\ x \\ y \end{Bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{Bmatrix} L_1 \\ L_2 \\ L_3 \end{Bmatrix}. \quad (3.54)$$

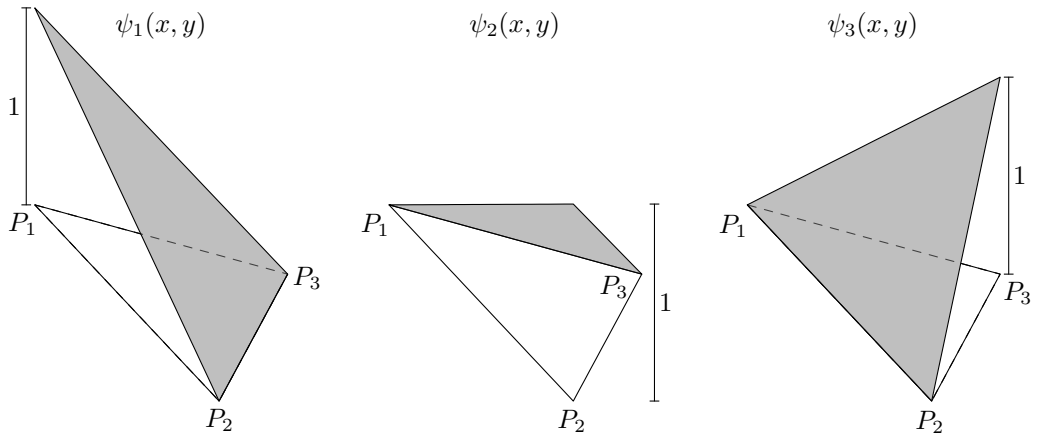


Fig. 3.4 First-order piecewise polynomial (affine) shape functions $\psi_1(x, y)$, $\psi_2(x, y)$ and $\psi_3(x, y)$.

With the shape functions written in local area coordinates (see Fig. 3.4)

$$\boldsymbol{\psi} = \begin{Bmatrix} \psi_1(x, y) & \psi_2(x, y) & \psi_3(x, y) \end{Bmatrix}^\top = \begin{Bmatrix} L_1 & L_2 & L_3 \end{Bmatrix}^\top,$$

the partial derivatives of ψ become very straightforward:

$$\frac{\partial \psi}{\partial x} = \begin{Bmatrix} B_1 & B_2 & B_3 \end{Bmatrix}^\top, \quad (3.55)$$

$$\frac{\partial \psi}{\partial y} = \begin{Bmatrix} C_1 & C_2 & C_3 \end{Bmatrix}^\top. \quad (3.56)$$

For the evaluation of the integral of a two-dimensional element, we can use

$$\iint L_1^\alpha L_2^\beta L_3^\gamma dx dy = \frac{\alpha! \beta! \gamma!}{(\alpha + \beta + \gamma + 2)!} 2\Delta_e. \quad (3.57)$$

For one-dimensional line element with length L this reduces to

$$\int L_1^\alpha L_2^\beta dS = \frac{\alpha! \beta!}{(\alpha + \beta + 1)!} L. \quad (3.58)$$

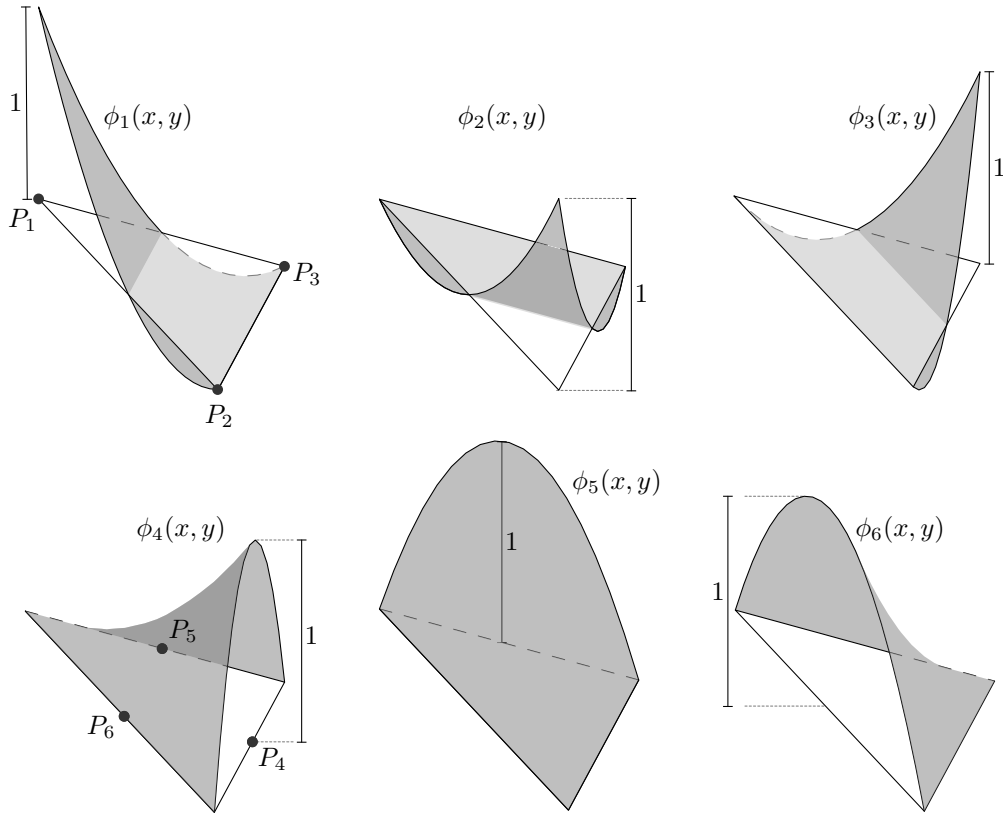


Fig. 3.5 Second-order piecewise polynomial (quadratic) shape functions $\phi_1(x, y), \dots, \phi_6(x, y)$.

The quadratic approximation (see Fig.3.5) of a test function $\phi(x, y)$ is defined using the area coordinates as

$$\phi(x, y) = \alpha_1 + \alpha_2 L_1 + \alpha_3 L_2 + \alpha_4 L_1^2 + \alpha_5 L_1 L_2 + \alpha_6 L_2^2. \quad (3.59)$$

For all six nodal values (vertices and midpoints in Fig. 3.3) of the test function can be written as

$$\begin{aligned} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \end{Bmatrix} &= \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1/2 & 0 & 0 & 1/4 \\ 1 & 1/2 & 0 & 1/4 & 0 & 0 \\ 1 & 1/2 & 1/2 & 1/4 & 1/4 & 1/4 \end{bmatrix} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{Bmatrix}, \\ \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{Bmatrix} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & -3 & 0 & 4 & 0 \\ 0 & -1 & -3 & 4 & 0 & 0 \\ 2 & 0 & 2 & 0 & -4 & 0 \\ 0 & 0 & 4 & -4 & -4 & 4 \\ 0 & 2 & 2 & -4 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \end{Bmatrix}. \end{aligned} \quad (3.60)$$

Substituting the coefficients in Eq. (3.59) with the above and using the relation in Eq. (3.52), the quadratic shape functions can be written as

$$\begin{aligned} \boldsymbol{\varphi} &= \left\{ \varphi_1 \quad \varphi_2 \quad \varphi_3 \quad \varphi_4 \quad \varphi_5 \quad \varphi_6 \right\}^\top, \\ &= \left\{ L_1(2L_1 - 1) \quad L_2(2L_2 - 1) \quad L_3(2L_3 - 1) \quad 4L_2L_3 \quad 4L_3L_1 \quad 4L_1L_2 \right\}^\top. \end{aligned} \quad (3.61)$$

3.2.5 Time Integrator

We have explained the spatial discretization of the Navier–Stokes Eq. (3.1) and the continuity Eq. (3.2) equation. Next, the time integrator must be chosen. We have chosen the backward differentiation formula of second-order (the same integrator we use for the particles, see Section 2.3). It is not only A-stable (able to obtain the solution of certain differential equations with transcendental solutions with arbitrary large time-step), but also L-stable (small perturbations are not damped out to zero³) [17]. Furthermore, it is implicit, i.e. there is no von Neumann stability condition which would limit the time-step for small spatial discretizations, an important aspect, as we can expect that our triangular grid inside the pore-space can become very small. The main purpose in choosing the finite element discretization was to obtain a continuous discretization for the whole space, to allow an exact discretization of the pore space and a smooth interpolation in the case

³In principle, the higher than second order BDF-methods can be imagined to damp out small perturbations, even if those are hydrodynamic instabilities which are physically significant.

where particles (boundaries) move, so that grid points have to be located. For the time integrator, other considerations become relevant. Some are related to the stability of the integrator itself, other are related to the stability with respect to the time-step and the spatial discretization, and a third aspect is for our simulation how well the integrator fits to the time integrator for the particles. For a first-order ordinary differential equation (ODE) $\dot{y} = f(y)$ we have

$$y_{n+1} - \frac{(1 + \omega_n)^2}{1 + 2\omega_n} y_n + \frac{\omega_n^2}{1 + 2\omega_n} y_{n-1} = +\tau_n \frac{1 + \omega_n}{1 + 2\omega_n} \dot{y}_{n+1}. \quad (3.62)$$

Eq. (3.62) is the version of BDF2 with variable step-size τ_n [32] where $\omega_n = \tau_n/\tau_{n-1}$ is the step-size ratio. When the ratio is $\omega_n = 1$, we obtain the version of BDF2 with fix step-size Eq. (2.52) as introduced in Section 2.3. Substituting the time derivative of the velocities $\dot{\mathbf{u}}$ in Eq. (3.33) with Eq. (3.62) gives us the DAE-form for the Navier–Stokes equation in the FEM-formulation with variable step-size version of BDF2 [28]:

$$\begin{aligned} & \begin{bmatrix} \frac{1 + 2\omega_n}{\tau_n(1 + \omega_n)} M + K + N(\mathbf{u}_{n+1}) & -C \\ C^\top & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{u}_{n+1} \\ \mathbf{P}_{n+1} \end{Bmatrix} \\ &= \begin{Bmatrix} M \left[\frac{1 + \omega_n}{\tau_n} \mathbf{u}_n - \frac{\omega_n^2}{\tau_n(1 + \omega_n)} \mathbf{u}_{n-1} \right] + \mathbf{f}_{n+1} \\ \mathbf{0} \end{Bmatrix}. \end{aligned} \quad (3.63)$$

Since Eq. (3.63) is implicit, the values of the \mathbf{u}_{n+1} and \mathbf{P}_{n+1} are computed using Newton–Raphson iteration with the appropriate predictor for the velocity via “generalized leapfrog”:

$$\mathbf{u}_{n+1}^P = \mathbf{u}_n + (1 + \omega_n) \tau_n \dot{\mathbf{u}}_n - \omega_n^2 (\mathbf{u}_n - \mathbf{u}_{n-1}), \quad (3.64)$$

as the initial guess. Then from the solution \mathbf{u}_{n+1} compute the required predictor data for the next time-step with

$$\dot{\mathbf{u}}_{n+1} = \frac{3\mathbf{u}_{n+1} - 4\mathbf{u}_n + \mathbf{u}_{n-1}}{2\tau_n}. \quad (3.65)$$

This shows that the method is indeed implicit, so that we are not bound by upper limits for the time-step due to the von Neumann stability condition. Using the local truncation error,

$$\mathbf{d}_n = \frac{(1 + \omega_n^{-1})^2}{1 + 3\omega_n^{-1} + 4\omega_n^{-2} + 2\omega_n^{-3}} (\mathbf{u}_{n+1} - \mathbf{u}_{n+1}^P), \quad (3.66)$$

we can compute the next time-step as

$$\tau_{n+1} = \tau_n \left(\frac{\varepsilon}{\max \|\mathbf{d}_n\|} \right)^{\frac{1}{3}}, \quad (3.67)$$

where the value of coefficient ε that we usually work with is 10^{-3} .

As the BDF2-method is not self-starting (i.e. at the first time-step, data from “before” the initial conditions are needed in Eq.(3.64)) and consistent initial conditions are also needed for the pressure, so that the incompressibility condition is not violated, we start our algorithm from a stationary state as obtained via Newton–Raphson iteration for the stationary Navier–Stokes equation. Alternative approaches have been proposed using one step of the trapezoid method [28], but this does not get around the problem of finding suitably consistent initial conditions for the pressures)

This approach of integrating out the time dimension via a solver for ordinary differential equations (ODE), instead of discretizing the time direction also via finite elements is sometimes referred to as “semi-discretization”. Generally, using ODE-solver for integrating out the time-evolution of partial differential equations is referred to as “method of lines” [33]. The advantage of using semi-discretizations is that the mature theories for the field of numerical ODE’s are available which classify the solvers with respect to accuracy and stability: For other approaches (Newmark-method), no such understanding exists, while incorporating finite elements in the time domain would enforce continuity, which would render the possibility of obtaining discontinuous pressures in time in the DAE-formulation practically useless. A final consideration in the choice of BDF2 was that the same time integrator can be used with with our polygonal discrete element simulation: That the integration process can be conducted in parallel reduces the risk that the integrator of one program part feeds noise into the other program part.

3.2.6 Newton–Raphson Method

The Newton–Raphson Method is a root-finding method which starts with an initial guess x_0 and iterates through the process until the difference between current approximated value and the previous one is small enough. For finding the root for a function $f(x)$, i.e. the value of x such that $f(x) = 0$, the Newton–Raphson

method (which can be derived from the Taylor expansion [34]) is given as

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad (3.68)$$

where the subscripts indicate the iteration steps (not time evolution). Note that the method requires the evaluation of both the function $f(x)$ and its derivative $f'(x)$ at x_i . The equation uses the slope of the function at x_i and extends its tangent line until it crosses zero. The next guess x_{i+1} is then set to be the abscissa of the zero crossing (see Fig. 3.6). This process is repeated with the next guess until the residue $(f(x_i)/f'(x_i) = x_i - x_{i+1})$ is lower than a certain tolerance. The method converges quadratically, i.e. in each iteration, the number of valid digits doubles [24].

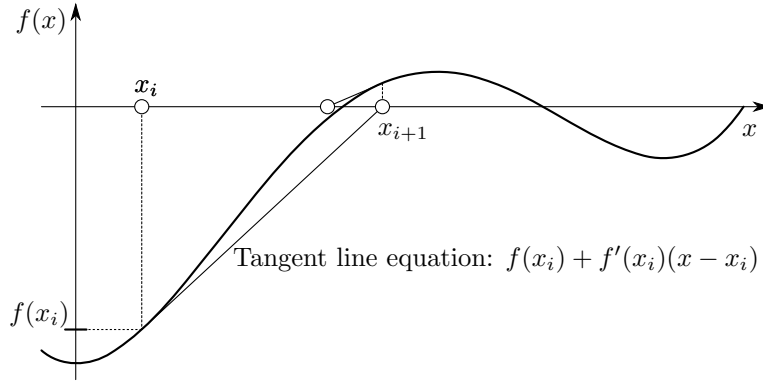


Fig. 3.6 Geometrical interpretation of the Newton–Raphson method: The local derivative $f'(x_i)$ of the function is extrapolated to find the next guess x_{i+1} .

The Newton–Raphson method can be extended higher dimensions and used to solved nonlinear systems of equations; instead of the derivative, the Jacobian is used. For a set of n simultaneous equations of the form,

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}, \quad (3.69)$$

where $\mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}) \ F_2(\mathbf{x}) \ \dots \ F_n(\mathbf{x})]^\top$ and the variables $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]^\top$, a set of linear equations for the residue vector $\Delta \mathbf{x}^{(i+1)}$ which is used to compute the next guess $\mathbf{x}^{(i+1)}$ is given as⁴

$$J(\mathbf{x}^{(i)}) \cdot \Delta \mathbf{x}^{(i)} = \mathbf{F}(\mathbf{x}^{(i)}), \quad (3.70)$$

where the superscripts indicate the iteration-step and J is the Jacobian of \mathbf{F} ,

$$J = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}. \quad (3.71)$$

⁴Detailed derivation can be found in [24].

With the Newton increment $\Delta \mathbf{x}^{(i)}$ from solving the linear algebraic equation Eq. (3.70), next guesses can be computed as

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \Delta \mathbf{x}^{(i)}. \quad (3.72)$$

This process is then repeated until the residue $\Delta \mathbf{x}^{(i)}$ is lower than a certain tolerance as in the one-dimensional case.

The Gâteaux derivate of a function F in the direction of an arbitrary increment Δx ,

$$F'_G(x)\Delta x = \lim_{\varepsilon \rightarrow 0} \frac{F(x + \varepsilon \Delta x) - F(x)}{\varepsilon}, \quad (3.73)$$

is used to define the left-hand side of Eq. (3.70) when applying Newton's method in solving the flow problem Eq. (3.63). We start with rewriting the Navier–Stokes Eq. (3.7) Eq. (3.7) and continuity Eq. (3.2) equations by substituting the time-derivative terms with the BDF2 formulation Eq. (3.62) as

$$\begin{aligned} H(u_{n+1}, v_{n+1}, P_{n+1}) &= \frac{1 + 2\omega_n}{\tau_n(1 + \omega_n)}u_{n+1} - \frac{1 + \omega_n}{\tau_n}u_n + \frac{\omega_n^2}{\tau_n(1 + \omega_n)}u_{n-1} \\ &+ u_{n+1}\frac{\partial u_{n+1}}{\partial x} + v_{n+1}\frac{\partial u_{n+1}}{\partial y} + \frac{\partial P_{n+1}}{\partial x} \\ &- \nu \left(\frac{\partial^2 u_{n+1}}{\partial x^2} + \frac{\partial^2 u_{n+1}}{\partial y^2} \right) - X, \end{aligned} \quad (3.74)$$

$$\begin{aligned} V(u_{n+1}, v_{n+1}, P_{n+1}) &= \frac{1 + 2\omega_n}{\tau_n(1 + \omega_n)}v_{n+1} - \frac{1 + \omega_n}{\tau_n}v_n + \frac{\omega_n^2}{\tau_n(1 + \omega_n)}v_{n-1} \\ &+ u_{n+1}\frac{\partial v_{n+1}}{\partial x} + v_{n+1}\frac{\partial v_{n+1}}{\partial y} + \frac{\partial P_{n+1}}{\partial y} \\ &- \nu \left(\frac{\partial^2 v_{n+1}}{\partial x^2} + \frac{\partial^2 v_{n+1}}{\partial y^2} \right) - Y, \end{aligned} \quad (3.75)$$

$$C(u_{n+1}, v_{n+1}) = \frac{\partial u_{n+1}}{\partial x} + \frac{\partial v_{n+1}}{\partial y}. \quad (3.76)$$

Using the definition at Eq. (3.73), the resulting Gâteaux derivative of the horizontal equation Eq. (3.74) in the above at the guesses $u^{(i)}$, $v^{(i)}$, $P^{(i)}$ and the Newton increment $\Delta u^{(i)}$, $\Delta v^{(i)}$, $\Delta P^{(i)}$ is

$$\begin{aligned} &H'_G(u^{(i)}, v^{(i)}, P^{(i)}) (\Delta u^{(i)}, \Delta v^{(i)}, \Delta P^{(i)}) \\ &= \lim_{\varepsilon \rightarrow 0} \frac{H(u^{(i)} + \varepsilon \Delta u^{(i)}, v^{(i)} + \varepsilon \Delta v^{(i)}, P^{(i)} + \varepsilon \Delta P^{(i)}) - H(u^{(i)}, v^{(i)}, P^{(i)})}{\varepsilon}, \\ &= \Omega_n \Delta u^{(i)} + \Delta u^{(i)} \frac{\partial u^{(i)}}{\partial x} + u^{(i)} \frac{\partial \Delta u^{(i)}}{\partial x} + \Delta v^{(i)} \frac{\partial u^{(i)}}{\partial y} + v^{(i)} \frac{\partial \Delta u^{(i)}}{\partial y} \end{aligned}$$

$$+ \frac{\partial \Delta P^{(i)}}{\partial x} + \nu \left(\frac{\partial^2 \Delta u^{(i)}}{\partial x^2} + \frac{\partial^2 \Delta u^{(i)}}{\partial y^2} \right), \quad (3.77)$$

where $\Omega_n = (1 + 2\omega_n)/(\tau_n(1 + \omega_n))$. Analogously, for the vertical and continuity equations we have

$$\begin{aligned} V'_G = & \Omega_n \Delta v^{(i)} + \Delta u^{(i)} \frac{\partial v^{(i)}}{\partial x} + u^{(i)} \frac{\partial \Delta v^{(i)}}{\partial x} + \Delta v^{(i)} \frac{\partial v^{(i)}}{\partial y} + v^{(i)} \frac{\partial \Delta v^{(i)}}{\partial y} \\ & + \frac{\partial \Delta P^{(i)}}{\partial y} + \nu \left(\frac{\partial^2 \Delta v^{(i)}}{\partial x^2} + \frac{\partial^2 \Delta v^{(i)}}{\partial y^2} \right), \end{aligned} \quad (3.78)$$

$$C'_G = \frac{\partial \Delta u^{(i)}}{\partial x} + \frac{\partial \Delta v^{(i)}}{\partial y}. \quad (3.79)$$

Using the same approach from Section 3.2.1, the weak forms for the Gâteaux derivatives are obtained as

$$\begin{aligned} \iint \phi^{(x)} \left(\Omega_n \Delta u^{(i)} + \Delta u^{(i)} \frac{\partial u^{(i)}}{\partial x} + u^{(i)} \frac{\partial \Delta u^{(i)}}{\partial x} + \Delta v^{(i)} \frac{\partial u^{(i)}}{\partial y} + v^{(i)} \frac{\partial \Delta u^{(i)}}{\partial y} \right. \\ \left. - \Delta P^{(i)} \frac{\partial \phi^{(x)}}{\partial x} + \nu \left(\frac{\partial \Delta u^{(i)}}{\partial x} \frac{\partial \phi^{(x)}}{\partial x} + \frac{\partial \Delta u^{(i)}}{\partial y} \frac{\partial \phi^{(x)}}{\partial y} \right) \right) dx dy, \end{aligned} \quad (3.80)$$

$$\begin{aligned} \iint \phi^{(y)} \left(\Omega_n \Delta v^{(i)} + \Delta u^{(i)} \frac{\partial v^{(i)}}{\partial x} + u^{(i)} \frac{\partial \Delta v^{(i)}}{\partial x} + \Delta v^{(i)} \frac{\partial v^{(i)}}{\partial y} + v^{(i)} \frac{\partial \Delta v^{(i)}}{\partial y} \right. \\ \left. - \Delta P^{(i)} \frac{\partial \phi^{(y)}}{\partial y} + \nu \left(\frac{\partial \Delta v^{(i)}}{\partial x} \frac{\partial \phi^{(y)}}{\partial x} + \frac{\partial \Delta v^{(i)}}{\partial y} \frac{\partial \phi^{(y)}}{\partial y} \right) \right) dx dy, \end{aligned} \quad (3.81)$$

$$\iint \psi \left(\frac{\partial \Delta u^{(i)}}{\partial x} + \frac{\partial \Delta u^{(i)}}{\partial y} \right) dx dy, \quad (3.82)$$

where $\phi^{(x)}$, $\phi^{(y)}$ and ψ are the test functions. Discretization of the weak form can be done by rewriting the guesses, Newton increments and test functions as linear combinations of the basis functions (as in Section 3.2.2),

$$\begin{aligned} u^{(i)} &= \boldsymbol{\varphi}_{(x)}^\top \mathbf{u}^{(i)}, & \Delta u^{(i)} &= \boldsymbol{\varphi}_{(x)}^\top \boldsymbol{\Delta} \mathbf{u}^{(i)}, & \phi^{(x)} &= \boldsymbol{\varphi}_{(x)}^\top \boldsymbol{\Phi}_{(x)}^*, \\ v^{(i)} &= \boldsymbol{\varphi}_{(y)}^\top \mathbf{v}^{(i)}, & \Delta v^{(i)} &= \boldsymbol{\varphi}_{(y)}^\top \boldsymbol{\Delta} \mathbf{v}^{(i)}, & \phi^{(y)} &= \boldsymbol{\varphi}_{(y)}^\top \boldsymbol{\Phi}_{(y)}^*, \\ P^{(i)} &= \boldsymbol{\psi}^\top \mathbf{P}^{(i)}, & \Delta P^{(i)} &= \boldsymbol{\psi}^\top \boldsymbol{\Delta} \mathbf{P}^{(i)}, & \psi &= \boldsymbol{\psi}^\top \boldsymbol{\Psi}^*. \end{aligned}$$

The resulting equations in matrix-vector representation are

$$\underbrace{\begin{bmatrix} \frac{1 + 2\omega_n}{\tau_n(1 + \omega_n)} M + K + L & -C \\ C^\top & 0 \end{bmatrix}}_{\equiv J} \underbrace{\begin{Bmatrix} \boldsymbol{\Delta} \mathbf{u}^{(i)} \\ \boldsymbol{\Delta} \mathbf{P}^{(i)} \end{Bmatrix}}_{\equiv \boldsymbol{\Delta} \mathbf{x}}, \quad (3.83)$$

where the partitioned matrices M , K and C are the same as those in Section 3.2.2, with

$$L = \begin{bmatrix} L_1 & L_{1,2} \\ L_{2,1} & L_2 \end{bmatrix}, \quad (3.84)$$

$$\Delta \mathbf{u}^{(i)} = \left\{ \left(\Delta \mathbf{u}_1^{(i)} \right)^\top \quad \left(\Delta \mathbf{u}_2^{(i)} \right)^\top \right\}^\top. \quad (3.85)$$

The matrices in L are defined as,

$$L_\alpha = \int \varphi_{(\alpha)} \left(\varphi_{(1)}^\top \mathbf{u}_1^{(i)} \frac{\partial \varphi_{(\alpha)}^\top}{\partial x_1} + \varphi_{(2)}^\top \mathbf{u}_2^{(i)} \frac{\partial \varphi_{(\alpha)}^\top}{\partial x_2} + \frac{\partial \varphi_{(\alpha)}^\top \mathbf{u}_\alpha^{(i)}}{\partial x_\alpha} \varphi_{(\alpha)}^\top \right), \quad (3.86)$$

$$L_{\alpha,\beta} = \int \varphi_{(\alpha)} \left(\frac{\partial \varphi_{(\alpha)}^\top \mathbf{u}_\alpha^{(i)}}{\partial x_\beta} \varphi_{(\beta)}^\top \right). \quad (3.87)$$

Note that $\mathbf{u}_1^{(i)}$ and $\mathbf{u}_2^{(i)}$ are the guesses for the Newton–Raphson method which are given, hence the formulation in Eq. (3.83) is linear. The resulting matrix is equivalent to the Jacobian J and the vector is equivalent to the residual vector $\Delta \mathbf{x} = \{(\Delta \mathbf{u}^{(i)})^\top (\Delta \mathbf{P}^{(i)})^\top\}^\top$ in Eq. (3.70). The Jacobian is also a sparse matrix i.e. (contains a large number of zero-elements, see Fig. 3.7) which allows us to make use of the functions for sparse matrices in MATLAB to store the Jacobian for saving memory and speed up the computing time. The right-hand side \mathbf{F} of Eq. (3.70) can be obtained by substituting \mathbf{u}_{n+1} and \mathbf{P}_{n+1} in the DAE-form for the Navier–Stokes equation in the FEM-formulation Eq. (3.63) with the guesses $\mathbf{u}^{(i)}$ and $\mathbf{P}^{(i)}$. Finally, the residual vector $\Delta \mathbf{x}$ is obtained by solving the linear algebraic equation

$$J \Delta \mathbf{x} = \mathbf{F}, \quad (3.88)$$

then with it, the next guesses are computed as in Eq. (3.72).

We have tested three different solvers on the linear systems of the form $Ax = b$ Eq. (3.88): The unsymmetric multifrontal LU factorization (UMFPACK) [35], the generalized minimal residual method (GMRES)⁵ and the biconjugate gradient stabilized method (BiCGSTAB(l)). All three methods are known for their efficiency when solving sparse linear systems and are available in MATLAB as built-in functions: `x = A\b`, `gmres()` and `bicgstab()` respectively. The Krylov-subspace-based iterative solvers (GMRES and BiCGSTAB(l)) are based on the idea of orthogonal vectors and applicable to non-symmetric matrices. In GMRES [36], a sequence of orthogonal vectors are computed, combined and formulated into

⁵There is an established GMRES-FEM-method, but it is for compressible flows.

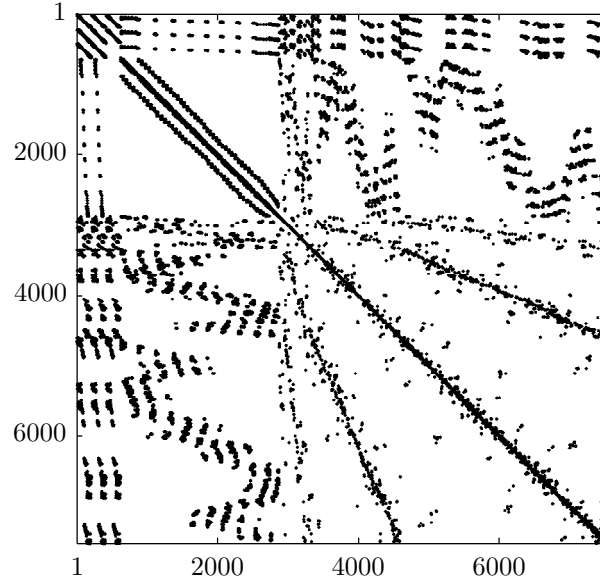


Fig. 3.7 The Jacobian matrix for the Newton–Raphson method when solving flow problem with FEM. Number of velocity nodes is 6576 and pressure nodes is 957. The non-zero elements are shown in black.

a least squares problem. The approximated solution of the linear systems x is obtained by minimizing the Euclidean norm of the residual in the least squares problem. On the other hand, $\text{BiCGSTAB}(l)$ [37] minimizes the residual constructed from the bi-conjugate gradient (Bi-CG) residuals and the l degree minimal residual polynomials. Compared to the original Bi-CG method, $\text{BiCGSTAB}(l)$ has much better convergence behavior and does not need the computation of the transpose matrix in Bi-CG. Fig. 3.8 shows our profiling results of the mentioned methods with Jacobian matrices of different number of nonzero elements in our simulation. The profiling results of GMRES are excluded from the graph to improve the readability as the time consumption of the GMRES is about 30 times higher than the $\text{BiCGSTAB}(l)$. We found that the Krylov-solvers turned out to be inefficient in our case as the velocities and the pressures in have different scaling. Bandwidth reduction (reordering of the nonzero elements closer to the diagonal) via the Cuthill–McKee algorithm [38] shows no significant improvement on the $\text{BiCGSTAB}(l)$ either.

Due to the different magnitudes between the velocities and the pressures, we are taking both the absolute error

$$\epsilon_{\text{abs}} = |\Delta \mathbf{x}|, \quad (3.89)$$

and relative error

$$\epsilon_{\text{rel}} = \left| \frac{\Delta \mathbf{x}}{\mathbf{x}^{(i+1)}} \right|, \quad (3.90)$$

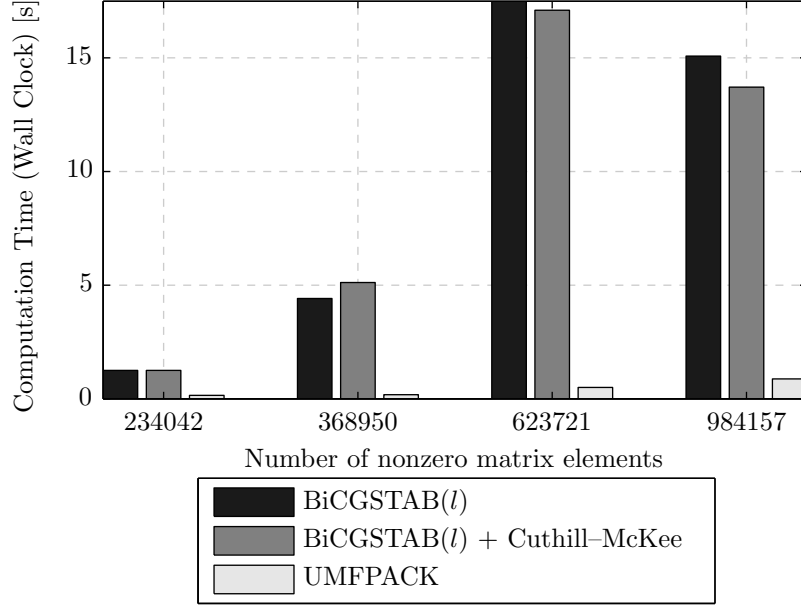


Fig. 3.8 Profiling results of solving the linear system $Ax = b$ with matrices A of different number of nonzero elements using BiCGSTAB(l) (black), BiCGSTAB(l) with Cuthill-McKee (gray) and UMFPACK (white).

into account when evaluating the residual for stopping the Newton-Raphson iterations. The iterations is terminated when all elements Δx_a in the residual vector are smaller than

$$\max(\tilde{\epsilon}_{\text{rel}} |x_a^{(i+1)}|, \tilde{\epsilon}_{\text{abs}}), \quad (3.91)$$

where the relative tolerance $\tilde{\epsilon}_{\text{rel}} = 10^{-6}$ and absolute tolerance $\tilde{\epsilon}_{\text{tol}} = 10^{-8}$. As the solution vector contains both velocities and pressures, and the relative magnitude of the pressures is about 20 times larger than of the velocities, in a future version a parameter will be introduced which will fix the relative magnitude of the tolerances between pressures and velocities.

The Newton-Raphson method does critically depend on the starting value: While for suitable starting values, they convergence is quadratic, for unsuitable starting values the iteration diverges very fast towards infinity. For the implementation in the BDF2-scheme, the predicted value of the velocities via “generalized leapfrog” Eq. (3.64) and the pressure from previous time-step are usually sufficient starting values. A diverging iteration has usually been the signature of other problems in the simulation (unsuitable or inconsistent grid, problematic boundary conditions or sudden, unphysical changes in the boundary conditions like abrupt force changes between the particles).

3.3 Automatic Mesh Generation

In order to combine the finite element method with the discrete element method, the very first step is to generate the triangular finite element meshes for the Taylor–Hood elements (Section 3.2.4) in the fluid phase around the particles. For the finite element method, triangular meshes should be close to equilateral and certainly not degenerate. A large angle in a triangle gives a bad approximation of the derivatives and thus a large error, so as a rule of thumb, all angles should be smaller than 135° [39]. To guarantee the quality of the mesh, we need a fully automatic mesh generator which is capable of generating valid grids (in the FEM-sense: space-filling and without obtuse angles) “on the fly” in every time-step over arbitrary domains. The information of the specified geometric boundary of the domain and the required distribution of the element size should be sufficient, without additional assumptions. This leads to our development of an automatic mesh generation using the constrained Delaunay triangulation and relaxation algorithm to optimize the meshes [40]. In principle, a mesh-refinement at high gradients of the flow-fields (velocities, pressure) would be desirable to improve the accuracy, but in the current version of the program, a mesh-generation for a flow around several hundred parameters was ambitious enough.

3.3.1 Constrained Delaunay Triangulation

Meshes generated by using Delaunay triangulation have the following properties:

1. To avoid degeneracies, a Delaunay triangle is constructed when only three grid points are co-circular.
2. The interior of a circumcircle associated with each Delaunay triangle contains no other grid point.
3. The domain formed by the resulting Delaunay triangles is a convex hull of the grid points.

These properties maximize the minimum angle for all triangles and tend to avoid triangles with sharp corners. Therefore, the Delaunay triangulation is a good tool to generate meshes for the finite element method as we need the triangular mesh to be as equilateral as possible. Dual to the Delaunay triangulation is the Voronoi lattice (see Fig. 3.9). The Voronoi cell around a mesh point is that region which is closer to that mesh point than to any other point. In turn, each vertex of a Voronoi cell

is the center of a circle on which all three corners of a triangle from the Delaunay triangulation are situated. In other words, each Voronoi vertex is associated with a particular Delaunay triangle.

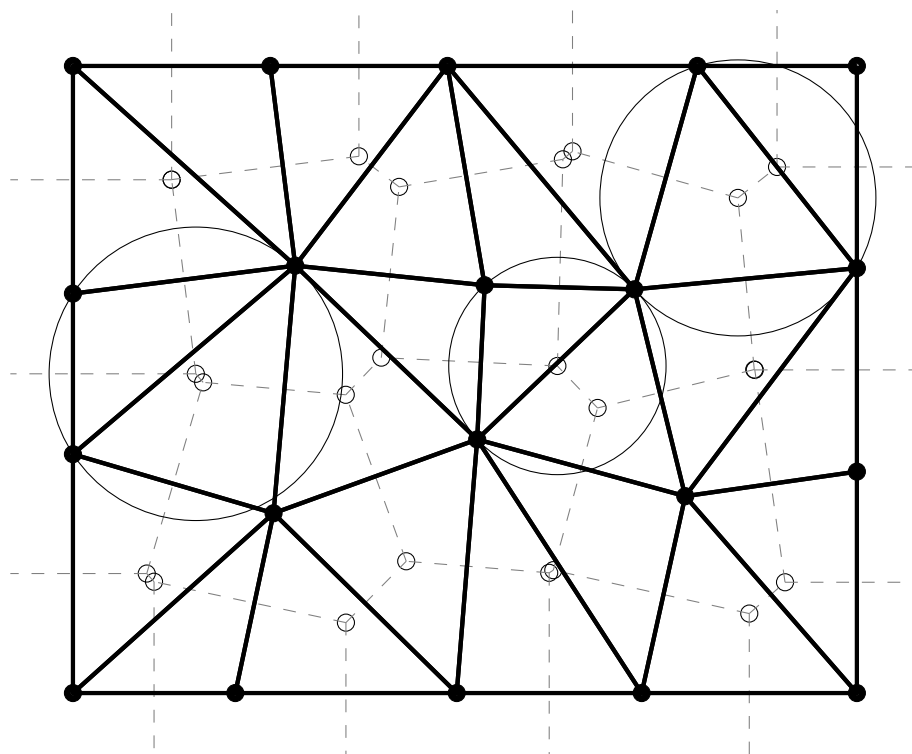


Fig. 3.9 Delaunay triangulation of set of a randomly positioned grid points (●) inside a rectangular domain. The corresponding Voronoi diagram is shown with $---$ and its vertices as ○. The interior of a circumcircle associated with each Delaunay triangle contains no other grid point.

However, the mathematically strict definition for the Delaunay triangulation is not suitable to generate the meshes needed for systems with particles inside the fluid, as the corresponding domain is not simply connected any more. Running the Delaunay-triangulation without precaution for our simulation would create triangles with edges which cut through neighboring particles (see Fig. 3.10 (a)). A Delaunay constructions which fulfills these additional conditions (but not necessarily all conditions of the mathematically exact Delaunay construction) is called constrained Delaunay triangulation. The introduction of the constraints leads to violations of the “properly” Delaunay triangulated meshes as the interior of a circumcircle of some Delaunay triangles may contain other grid points as shown in Fig. 3.10 (b).

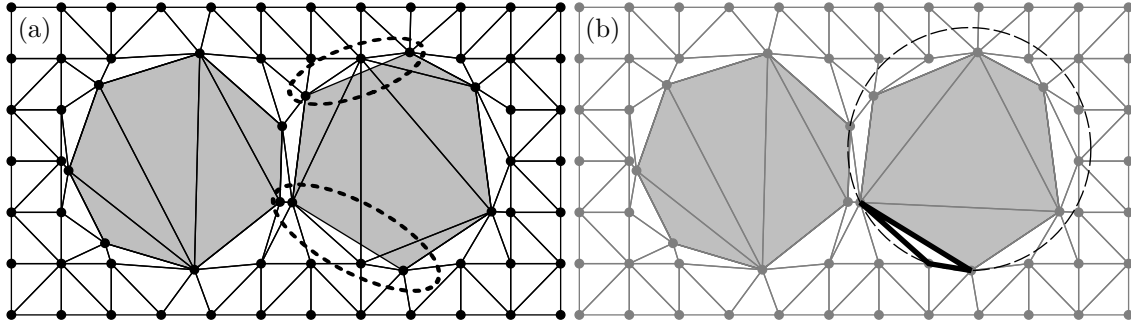


Fig. 3.10 Mesh generation around particles with (a) mathematically exact Delaunay triangulation, where triangles are cutting through a particle and (b) constrained Delaunay triangulation: Introduction of the constraints leads to not so properly Delaunay triangulated meshes, i.e. the circumcircle of a triangle (in thick lines) contains other grid points.

3.3.2 Relaxation Algorithm

In order to process the constrained Delaunay triangles in Section 3.3.1 even further, we are going to introduce a method to systematically improve the quality of the meshes with a “relaxation algorithm” where edges of a triangular mesh are treated as linear springs which are under compression (if the edge is shorter than the average of the three edge lengths) or tension (if an edge is longer than the average of the three edge lengths) as long as the triangles are not equilateral and which try to relax towards an equilibrium of an equilateral shape.

Relaxation Algorithm in One Dimension

We start with the explanation with a one-dimensional case where points x_1, x_2, \dots, x_n are randomly positioned on a straight line as shown in Fig.3.11. To obtain equidistant points, we introduce linear springs with spring constant k . The force F_i between neighboring points depends on the average distance between the points, equal to

$$\bar{x} = \frac{1}{m-1} \sum_{i=1}^{n-1} (x_{i+1} - x_i). \quad (3.92)$$

Then one obtains Hooke’s law for a linear chain for x_2, \dots, x_{n-1}

$$F_i = k((x_{i+1} - x_i) - \bar{x} - [(x_i - x_{i-1}) - \bar{x}]), \quad (3.93)$$

which means that springs are under compression if their endpoints are closer than \bar{x} and under tension if their endpoints are at a larger distance than \bar{x} . This reduces

to

$$\ddot{x}_i = x_{i+1} + x_{i-1} - 2x_i, \quad (3.94)$$

with spring constant $k = 1$ and unit mass. Accordingly, the x_i will experience forces from the left and the right, and the sum of this forces will give the direction towards the equilibrium positions. If such a grid-point $x_i(t)$ oscillates around an equilibrium position x_i^0 , x_i^0 is the time-independent approximation to $x_i(t)$, which is equivalent to the zero order. If there is no equilibrium, the zero order approximation will change with time and move towards the actual equilibrium position. Zeroth-order time-integrators can be obtained by manipulating higher order integrators and introducing controlled errors, e.g. for the second-order Verlet method [41] which is convenient to obtain the position vector $\mathbf{r}(t)$ from position-dependent accelerations $\ddot{\mathbf{r}}$ as

$$\mathbf{r}_{n+1} = 2\mathbf{r}_n - \mathbf{r}_{n-1} + \tau^2 \ddot{\mathbf{r}}_n + \mathcal{O}(\tau^4). \quad (3.95)$$

Then, we obtain a zeroth-order integrator by introducing a first-order error

$$\mathbf{r}_n = \mathbf{r}_{n-1} + \mathcal{O}(\tau), \quad (3.96)$$

as

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \tau^2 \ddot{\mathbf{r}}_n + \mathcal{O}(\tau). \quad (3.97)$$

As can be seen in Fig.3.11, this integrator is able to obtain the equilibria of mechanical systems described in Eq. (3.94).

The convergence can be derived in the following way: For a particle positioned at $x_n = x_0 + \rho$ in a potential

$$\Psi(\rho) = \psi_0 + \psi_2 \rho^2 + \dots, \quad (3.98)$$

where $\psi_2 > 0$ and the equilibrium is at x_0 , the acceleration of the particle (for unit mass) is

$$\ddot{x}(\rho) = -\nabla \Psi(\rho) = -2\psi_2 \rho + \dots. \quad (3.99)$$

Substituting the above equations into the zeroth-order approximation Eq. (3.97)

$$\begin{aligned} x_{n+1} &= x_0 + \rho + \tau^2 (-2\psi_2 \rho + \dots) \\ &\approx x_0 + \rho \underbrace{(1 - 2\psi_2 \tau^2)}_{-1 < \dots < 1}, \end{aligned} \quad (3.100)$$

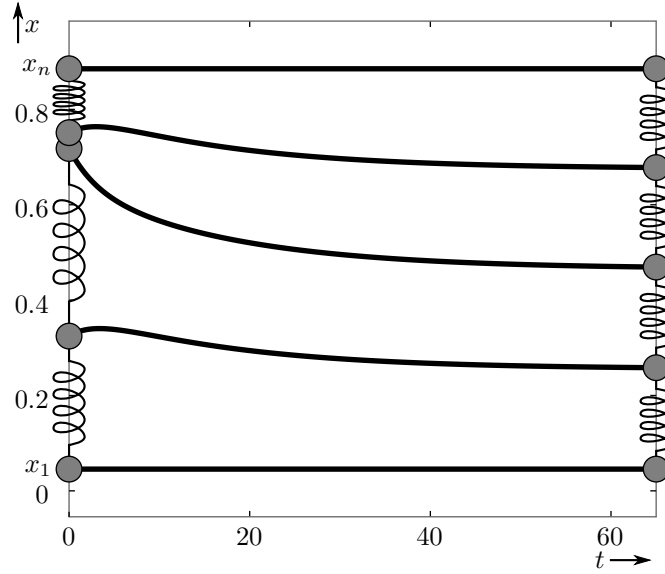


Fig. 3.11 Convergence of randomly spaced n points on the y -axis with interactions of linear springs between neighboring points towards the equilibrium (equidistant spacing) using the integrator from Eq. (3.97). The endpoints x_1 and x_n are fixed so that their position will not change.

shows us that for sufficiently small time-step $0 < \tau < 1/\sqrt{\psi_2}$, the resulting the new position x_{n+1} will be closer to the equilibrium x_0 than the old position $x_n = x_0 + \rho$. Only for the force equilibrium at $\rho = 0$, no force acts on the particle and the particle will remain there. On the other hand, unphysical oscillations from one step to the next will occur if the time-step is too large and the amplitudes may even diverge beyond a certain critical time-step.

Relaxation Algorithm in Two Dimensions

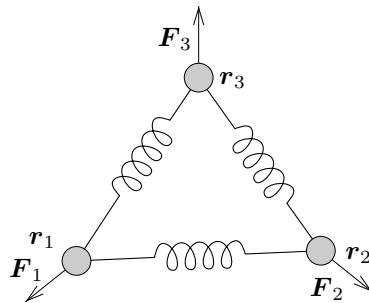


Fig. 3.12 Edges of triangular mesh treated as linear spring for relaxation algorithm. \mathbf{r}_1 , \mathbf{r}_2 and \mathbf{r}_3 are the position vectors of the vertices.

For the two-dimensional case which should be used for the grid-relaxation, additionally the directions must be introduced. Again we want to use the zeroth-order method to relax the vertices of a triangular mesh to obtain a mesh with

triangles which are as close to equilateral as possible. As shown in Fig. 3.12, the edges of the triangle are treated as linear spring. Using the oriented particle connections $\mathbf{l}_{12} = \mathbf{r}_1 - \mathbf{r}_2$ and $\mathbf{l}_{13} = \mathbf{r}_1 - \mathbf{r}_3$, the force \mathbf{F}_1 on \mathbf{r}_1 can be written as

$$\mathbf{F}_1 = - \left(|\mathbf{l}_{12}| - \tilde{l} \right) \frac{\mathbf{l}_{12}}{|\mathbf{l}_{12}|} - \left(|\mathbf{l}_{13}| - \tilde{l} \right) \frac{\mathbf{l}_{13}}{|\mathbf{l}_{13}|}, \quad (3.101)$$

where \tilde{l} is the arithmetic mean length of all three of the triangles edges. Like in the one-dimensional case, point \mathbf{r}_1 will experience an attracting force from \mathbf{r}_2 or \mathbf{r}_3 if the distance between them is larger than \tilde{l} , and a repulsive force if the distance is smaller than \tilde{l} . With the force definition and the zeroth-order method in Eq. (3.97), we can run the relaxation algorithm on triangular meshes to improve the quality of the meshes in a systematic way. During the process, the constrained Delaunay triangulation can be performed again after some relaxation steps in order to prevent triangles from overlapping with the boundaries. The actual time consumption of our remeshing algorithm is not a concern: For a fluid simulation around granular particles, the fluid part (in particular, in our case, the solver) can be expected to take much more computer time than the remeshing or the relaxation algorithm. Moreover, the time consumption may increase as the introduction of the particles introduces an additional smaller timescale with the particle collision time, compared to the fluid alone, at least for high viscosities. On the other hand, simulations nevertheless indicate that for the Reynolds numbers achievable, the collision time for simulations with fluid were larger than for particle simulations without fluid.

Fig. 3.13 (b) shows the resulting meshes by applying the relaxation algorithm on the constrained Delaunay triangles in Fig. 3.10 (b). One can use the following equation to measure the quality of a triangle [42, 43]

$$q = \frac{4\sqrt{3}A}{l_{12}^2 + l_{23}^2 + l_{31}^2}, \quad (3.102)$$

where A is the area of the triangle and l_{12} , l_{23} and l_{31} are the length of the triangle's edges. While an equilateral triangle ($l_{12} = l_{23} = l_{31}$) has $q = 1$, $q > 0.6$ is still considered to be of good quality. The application of the relaxation algorithm in Fig. 3.13 (b) reduces the number of bad quality meshes from 17% to 6%. Also the number of meshes close to equilateral triangle ($q > 0.95$) increases from 7% to 38%. The remaining triangles are near or connected to the boundaries have fixed vertices therefore not much can be done geometrically to improve the quality via relaxation algorithm.

One can extend the relaxation algorithm even further by stiffening the springs in

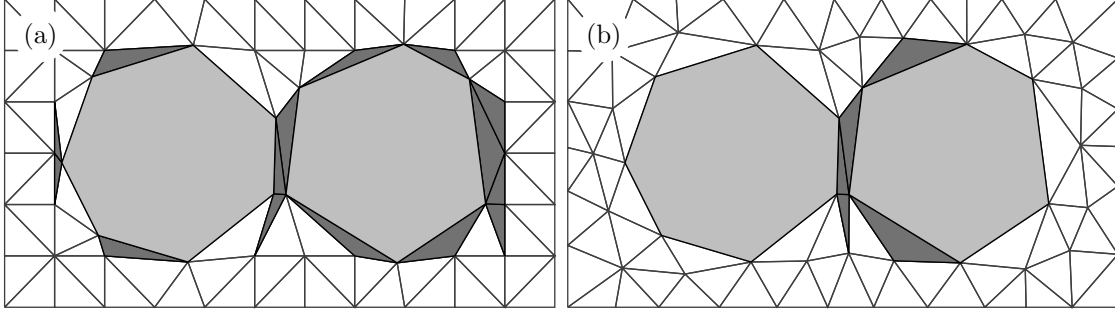


Fig. 3.13 (a) Constrained Delaunay triangulation and (b) the application of the relaxation algorithm. Bad quality meshes ($q \leq 0.6$) are shown in dark gray.

the regions where higher accuracies are needed to obtain finer meshes. Fig. 3.14 shows result of mesh adaption due to an external field. The region with light shading indicates high amplitude (region where higher accuracy is needed) while the dark region indicates low amplitude. By setting the values of the spring constants proportional to the field's amplitude, we were able to generate finer meshes in the field with high amplitude from an ordinary square grid systematically and automatically and improve the resolution and accuracy. While such a mesh-refinement has turned out to be very practicable, for performance reasons we would be equally interested in mesh coarsening in regions where the field gradients are small, so that the number of grid points and the computer time could be reduced considerably. Unfortunately, we have currently no algorithm available for that purpose.

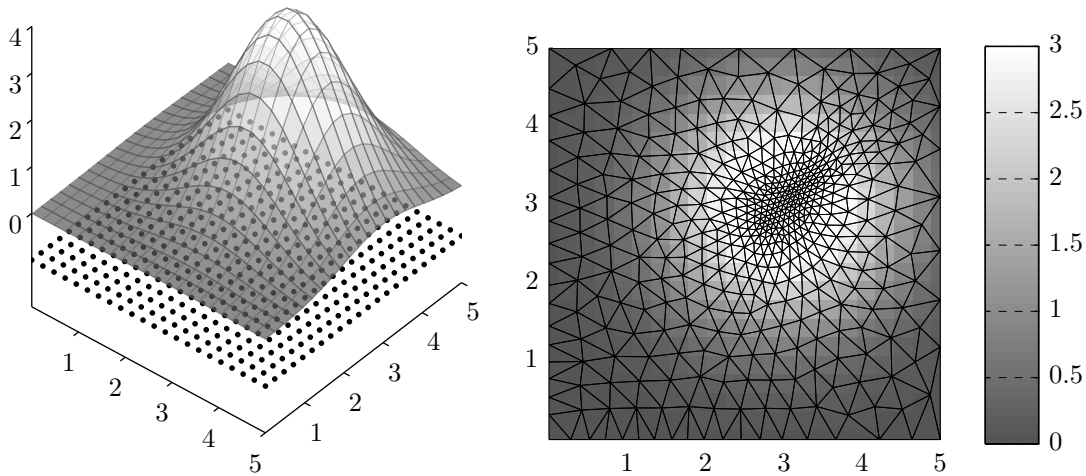


Fig. 3.14 Grid adaption (right) by applying the relaxation algorithm on a triangular mesh of a square grid with a Gaussian potential (left).

Chapter 4

Simulation of Flow Problems with Free Boundaries

Before we put the granular particles in our flow simulation, we will use our finite element code to simulation flow problem with free surface to “test the waters”. As a minimum requirement, one can consider that the flow field is computed so that the volume does not increase and flow velocities are correctly reproduced. The motion of the fluid surface is obtained by integrating out the velocities obtained from the FEM-simulation with integrators for ordinary differential equations (ODEs) without the need of additional data structures or interpolated mesh points as in the front tracking method. This approach minimizes the computational overhead for the surface, both with respect to data structures and to mathematical formalism [44]. As verification, we compare water column simulations with the lubrication approximation by Huppert [45] for high viscosity fluid and with the experimental data for water by Martin and Moyce [1] for low viscosity fluid.

4.1 Philosophy of the Surface Computation

While there is a mathematically developed theory for continua and their discretizations, the treatment of the respective surfaces is in a much less mature state. All standard solutions which deal with surfaces in fluid flow have their drawbacks, as the computational complexity is increased considerably compared to the flow problem without surfaces. Among the first approaches was the marker-and-cell method [27, 46] for finite differences where the introduction of the “markers” and the respective boundary conditions leads to totally new mathematical entities (interpolations, new mesh points) which are absent in the original flow problem

and the respective simulation approach. Several standard methods have evolved which are used to deal with surfaces which we nevertheless will try to avoid: Level set methods [47] have become standard tools to smoothly interpolate data which are available only on discrete meshpoints of (usually) rectangular grids. Nevertheless, these methods in computer science with a relatively weak focus on symmetries like Galilean invariance or isotropy which are relevant for mechanical problems. We cannot be sure that the solutions would be the same under a rotation of the coordinate system (which is partly due to the underlying grid), nor does a computer-graphics based smooth interpolation guarantee that the physical properties of the surface are consistent with the underlying fluid. For finite element simulations, the modeling of the surface with an additional advection equation has been proposed [48], but the mathematical complexity of introducing an additional partial differential equation to the Navier–Stokes equations may be too demanding for many users both with respect to the additional mathematical effort and the underlying mathematical assumptions.

As we have a finite element simulation described in Chapter 3 which allows to restructure and remesh the grid in every time-step, we would rather stick with our FEM-grid without introducing additional data structures for a variety of reasons:

“Economy of thought”: “Ockham’s razor” is the principle which states that unnecessary principles should not be introduced to solve a problem: When our grid already defines the boundary, why introduce another “grid” for the boundary? In particular, we are rather averse to introduce an entity which does not exist in nature: What is the surface of the fluid should simultaneously be its physical boundary, so the surface information should also be sufficient to model the boundary without additional theoretical assumptions or models.

Mechanic impedance: When we introduce, additionally to the boundary, a geometric entity with which the fluid interacts, there is a risk that this will alter the mechanic impedance (the way and speed that mechanic signals propagate over the boundary, or how they are reflected there) without the possibility to control or evaluate this effect.

Practicability: We also want to simulate porous media with particles and many thousand fluid interstices etc.. Introducing such an immense number of additional boundaries, including the corresponding overlaps or intersections may lead to non-unique algorithmic choice.

Analogy: Finite elements methods in structural mechanics don't need additional data structures to describe the surfaces. So shouldn't we be able to do without additional constructs als in fluid modeling?

4.2 Surface Modeling via ODE-Integrators

Integrators for ordinary differential integrators (ODEs) used in mechanics are basically methods to predict from positions $x_i(t)$ and velocities $v_i(t)$ the values for $x_i(t + \tau)$. From this standpoint, it should be sufficient to chose a suitably chosen ODE-solver, together with the positions of the surface $x_i(t)$ and the velocities $v_i(t)$ obtained from the FEM-code to obtain the new surface positions $x_i(t)$. Since we do not want to deform our P_2P_1 -elements into something with curved boundaries, we move only the corners of the triangles and interpolate the center points of the edges accordingly (see Fig. 4.1).

4.2.1 Choosing the Integrators

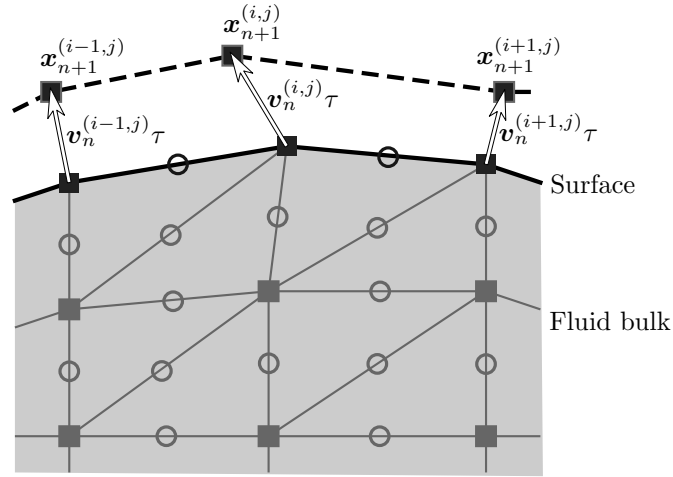


Fig. 4.1 Free surface modeling: moving the surface of the fluid using the obtained velocities from the finite element method. In order to avoid deformation in the P_2P_1 -elements, only the corners ■ of the triangles will be moved, and the midpoints ○ are interpolated accordingly.

The semi-discrete implementation is already implemented in our FEM-equations i.e. a FEM-grid with finite difference formulation of the time-evolution. Advancing the surface points of the FEM-grid with ODE-integrators will change the grid points in every time-step, and also make changes in the neighboring layers of points necessary, but as we have already opted for an approach which allows to restructure

the mesh in every time-step, there results no additional burden from this approach. In the simplest case, with the position of the surface at the discrete time-step n is given for the lattice site (i, j) as $\mathbf{x}_n^{(i,j)}$, and the velocity obtained by our FEM-procedure on the free surface as $\mathbf{v}_n^{(i,j)}$ (for a sketch, see Fig. 4.1) we could compute the new position of the lattice point via an Euler-integration for a timestep τ as

$$\mathbf{x}_{n+1}^{(i,j)} = \mathbf{x}_n^{(i,j)} + \mathbf{v}_n^{(i,j)}\tau. \quad (4.1)$$

Of course, due to the bad stability- and accuracy-properties of the Euler-method, it is advisable to look for ODE-solvers with better numerical properties, i.e. essentially for higher order solution alternatives for Eq. (4.1).

Since we have to make use of the velocities at the intervals which are determined by the BDF2-integration of the internal flow, we have the velocity available only at discrete times. Therefore, one-step methods (i.e. Runge–Kutta-type methods which use several function evaluations per time-step) are not suitable. Multi-step-methods, on the other hand, can compute new values x_{n+1}^{ij} from the old values $x_n^{ij}, x_{n-1}^{ij}, x_{n-2}^{ij}, \dots$ and the corresponding derivatives $v_n^{ij}, v_{n-1}^{ij}, v_{n-2}^{ij}, \dots$ (Our BDF2-integrator is also a multistep-method, but due to the implicitness of the equations, it is not suitable for the surface computation.) For changing positions of the $x \dots$ near the surface, interpolation-methods for retracting surfaces and extrapolation for advancing will become necessary.¹ Due to the imponderables with respect to interpolation, and because the $v \dots$ are computed with limited accuracy order anyway, a too higher order for the integrator of the surfaces is not suitable: A high order of the integrator, with larger errors (noise) on the input data will lead to a large error (or noise) for the surface computation. As a test-case, we chose to investigate gravity waves: An initial sinusoidal surface wave of not too large amplitude (and therefore, non-turbulent flow) evolves with time in a domain which is bounded by walls on the left and the right. Because the evolution is intuitively clear (viscous damping of the amplitude, vanishing amplitude on the walls), one can detect problems with the algorithms (perturbations spreading out from the near-wall region, unphysical wriggles forming on the surface, etc.) immediately from the graphical output. The effect of the integrator for gravitation waves in Fig. 4.2 was investigated and we found that the second-order Adams–Bashforth (AB2) method among various multi-step methods turned out to have the best volume conservation

¹Interpolation methods for varying timesteps instead of the equidistant $n, n+1, \dots$ are already available as standard tools in the field of numerical ODE's.

Fig. 4.3[49]

$$\mathbf{x}_{n+1}^{(i,j)} = \mathbf{x}_n^{(i,j)} + \tau \left(\frac{3}{2} \mathbf{v}_n^{(i,j)} - \frac{1}{2} \mathbf{v}_{n-1}^{(i,j)} \right). \quad (4.2)$$

Both Euler Eq. (4.1) and third-order methods gave worse results: The Euler method, because it lacks accuracy and stability i.e. the order is lower than the accuracy with which the velocities are obtained, and the third-order methods probably because the accuracy order is the same as the order of the noise for the velocity computation, so the noise from the velocities is propagates as if were a physical signal. We refrained from rescaling the volume etc. to see the immediately conservation ability for the integrators.

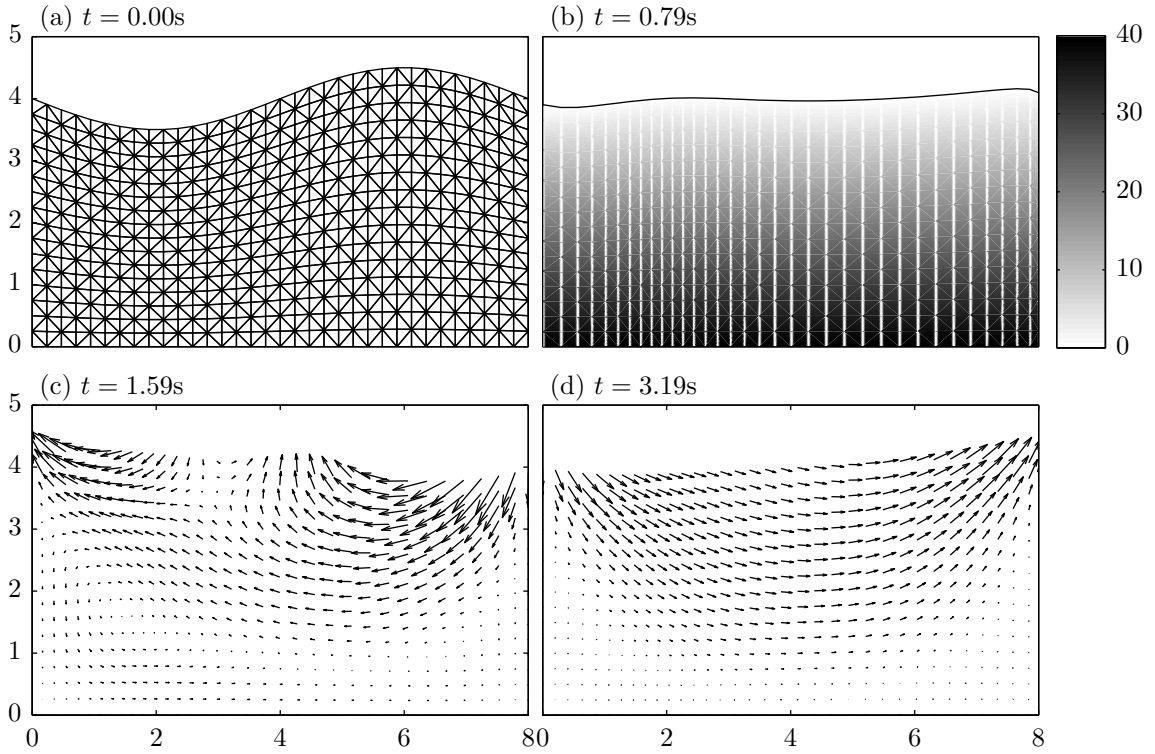


Fig. 4.2 Snapshots from the time evolution of the gravity wave with the triangular meshes in (a), pressure distribution in (b), flow velocities in (c) and (d).

4.2.2 Using Interpolated Velocities

We have up to now only discussed how to obtain the new positions from the “old” velocities using the either the Euler method eqs. Eq. (4.1) or the AB2 Eq. (4.2). We also have to discuss at which position we evaluate the velocities. Since with the FEM-simulation an we have Eulerian formulation where the evaluation of the velocities takes place at given grid-points, we should not just use the old positions at different time-steps: As the philosophy of our method is, rather than trace the

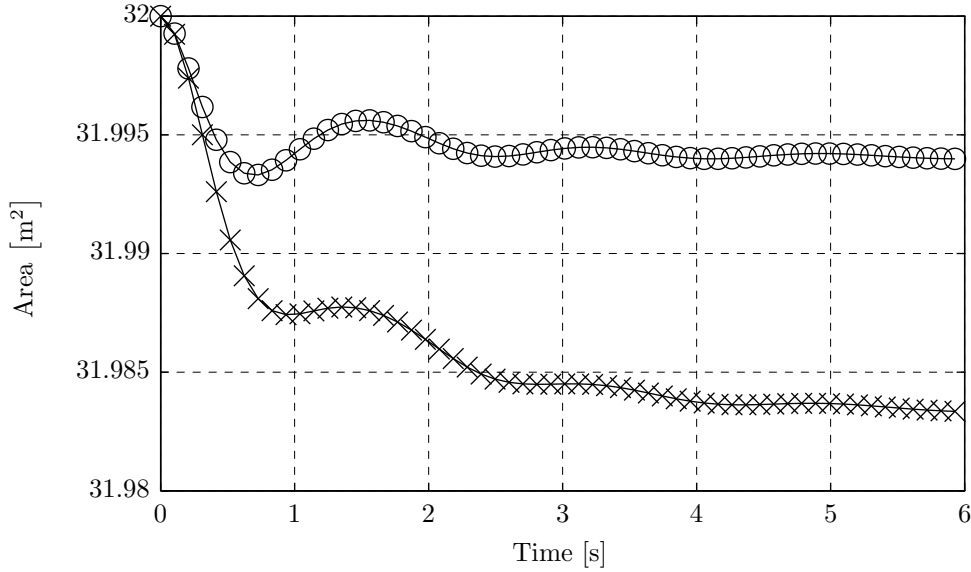


Fig. 4.3 Time evolution of the volume for the surface integration with Euler \times and Adams–Bashforth of second order \circ . The exact value is $32[\text{m}^2]$. When the motion of the surfaces slows down, the increase of the error also slows down. Symbols are not drawn for every time-step to keep the figure readable.

movement of the lattice points in a Lagrangian way, to predict the new position of the surface nodes \mathbf{x}_{n+1} , we choose for the Adams–Bashforth method

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \tau \left\{ \frac{3}{2} \mathbf{v}_n(\mathbf{x}_n) - \frac{1}{2} \mathbf{v}_{n-1}(\mathbf{x}_n) \right\}, \quad (4.3)$$

i.e. using both current \mathbf{v}_n and previous flow velocities \mathbf{v}_{n-1} at the the current position of the grid-point \mathbf{x}_n . If the surface grid-point \mathbf{x}_n is inside the fluid domain of the previous step ($n-1$), e.g. the surface is falling, we obtain the flow velocity of the previous step at that position $\mathbf{v}_{n-1}(\mathbf{x}_n)$ via interpolation. This interpolation from the FEM-method inside the domain is unique and smooth. Unfortunately, if the grid-point is located a the region where there was no fluid at the previous time-step, we have to abandon the use of Eq. (4.3) and just use the velocities of the grid point at the previous time-step in AB2 (see Fig. 4.4),

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \tau \left\{ \frac{3}{2} \mathbf{v}_n(\mathbf{x}_n) - \frac{1}{2} \mathbf{v}_{n-1}(\mathbf{x}_{n-1}) \right\}, \quad (4.4)$$

i.e. instead of the velocity of the flow field at the position, we use the velocity of the moving material point.

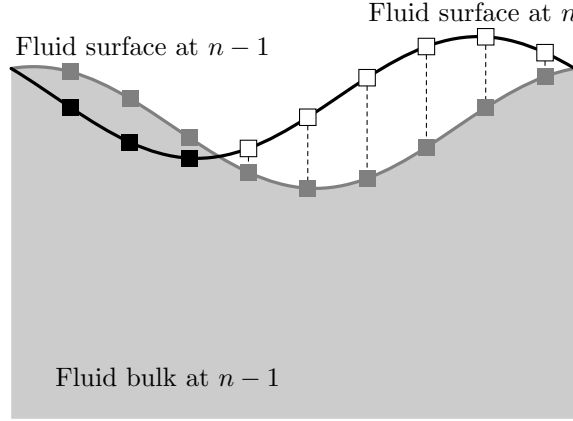


Fig. 4.4 Choosing the velocity data for the ODE-integrator to compute the surface node for the next time-step ($n + 1$): Using the interpolated value $\mathbf{v}_{n-1}(\mathbf{x}_n)$ if the current surface node \blacksquare is inside the fluid domain of previous step ($n - 1$). If the current surface node \square is located at the region where there was no fluid at the previous time-step, use the velocities of the grid point at the previous time-step \blacksquare .

4.3 Boundary Conditions

While the common boundary conditions in fluid mechanics are no-slip conditions, these conditions are obviously invalid at the fluid surface, else it would not be possible that a fluid meniscus which touches a boundary rises or falls. The necessity for the modification in comparison to conventional fluid simulations becomes obvious in Fig. 4.5. The fluid flow on the wall inside the boundaries of the fluid is indeed implemented as no-slip boundary condition (\circ in Fig. 4.5). However, to model realistically the movement of the water front near a solid boundary, we have to give

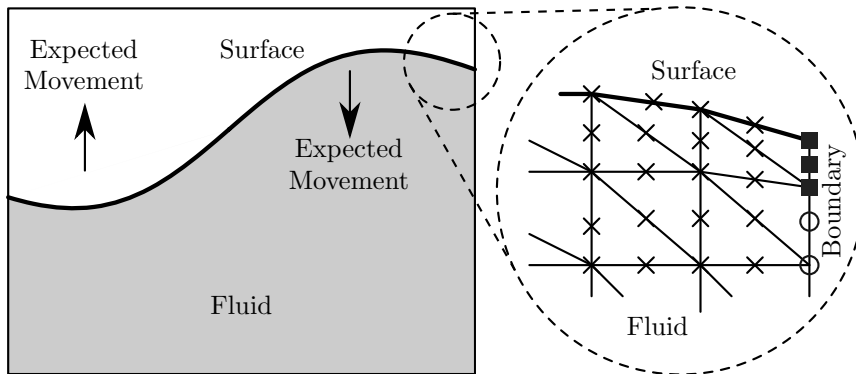


Fig. 4.5 Treatment of boundary conditions near the between the surface and the solid wall: Enforcing no-slip condition (normal and tangential velocities set to zero) on the boundaries nodes \circ except the ones of the outmost element. For the outmost element \blacksquare , the normal nodal velocity is set to zero while the tangential velocity is computed. Pressures on the surface are set to zero. For other nodes which do not touch the wall (including the surface nodes) \times the velocities are computed.

up the conventional no-slip condition in tangential direction between fluids and solids near the surface: Obviously, the water surface must be allowed to move. That does not mean that the water flow occurs directly at the the boundary: It is much more easy to imagine that fluid moves from the free surface towards the boundary as shown in Fig. 4.6 (or, for falling water levels, from the boundary towards the free surface), as in Fig. 4.2 (c)). When one fills up a glass with water, the fluid meniscus rises with a finite velocity, proportional to the filling rate. For the boundary, this manifests with a finite vertical velocity, incompatible with non-slip periodic boundaries. While non-slip boundary conditions are a reasonable extrapolation for macroscopic volume elements towards the boundary (vanishing momentum transfer), they are unsuitable to describe surface wetting phenomena (finite material transport). While the normal component of the flow velocity on the wall is still zero, the tangential velocity component on the fluid surface which contacts the wall must be computed from Eq. (3.63), not be assigned. An element of the finite element method may represent, depending on the problem, either a finite domain or an infinitesimal domain of the problem. Accordingly, the velocities of more or less nodes have to be computed, and there a priori three choice to compute the tangential velocity:

1. for the outmost node of the outmost element,
2. for the outmost element (for our P_2 -discretization of the velocities, that would mean three nodes) or
3. for the nodes of several elements near the boundary.

As for the actual issue with the finite element is the condition of the resulting matrices, which must be inverted to complete the Newton–Raphson-step, the

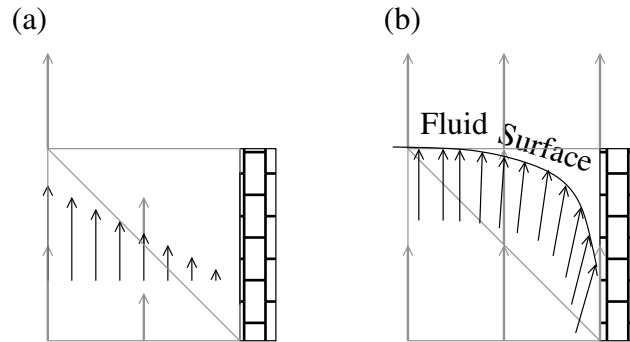


Fig. 4.6 Physical (black arrows) and finite element flow (gray arrows). Comparison between flow at at no-slip boundary (a) and the rising of the surface near a no-slip boundary (b). While on length-scales smaller than the mesh size, there may be microscopically more complicated flow patterns as in (b), for macroscopic modeling this leads to the computation of the tangential velocity.

issue has to be resolved according to numerical practicability rather than due to theoretical argument. For large viscosities ($\nu = 10^{-1} [\text{m}^2/\text{s}]$), it is enough to move only the outmost lattice point, i.e. the contact point between surface and boundary. For small viscosities, it turned out that the wave front propagation needed the computation of the tangential velocities of at least one element (three nodes) else the simulation became unstable. As far as the inclusion of more nodes is concerned: As the results for one and two elements were consistent, we stuck to the computation of the tangential velocities for a single element. For a free surface in the absence of surface tension, every node on the surface is force-free so the pressures are set to zero as boundary conditions. The initial pressure profile we computed with zero pressure on the surface, which in the fluid bulk lead to a hydrostatic pressure distribution, where the pressure was higher below “hills”, and lower under “valleys” of the wave.

4.4 Simulation of the Collapse of a Water Column

Beyond the volume conservation, which is basically only a consistency criterion, the true physicality of the simulation approach will manifest in the actual spreading of a fluid front. For the verification of our algorithm, we choose to simulate the collapse of a water column, i.e. the time evolution of a water-“step” which is a popular test-case, especially for Lagrangian (particle-based) methods. As discussed in Section 4.3, for a free surface in the absence of surface tension therefore for every node on the surface (including the vertical face), the pressures are set to zero. Setting the pressures only on the horizontal face to zero leads to hydrostatic pressure profile which would be the same as for a bounding wall, therefore inhibit any time evolution of the water column (see Fig. 4.7). This approach makes it easy to control the time for the “breaking of the dam”, while is experimentally rather difficult to remove a wall without disturbing the neighboring fluid.

4.4.1 Water Column with High Viscosity

We start to discuss the problem for high viscosity, where analytical reference data are available and the computation is more stable and less computationally demanding. Huppert [45] derived the dependence on time t of the advance of the front Z in the lubrication approximation as

$$Z(t) = 1.411 \left(\frac{gq^3}{3\nu} \right)^{\frac{1}{5}} t^{\frac{1}{5}} \quad (4.5)$$

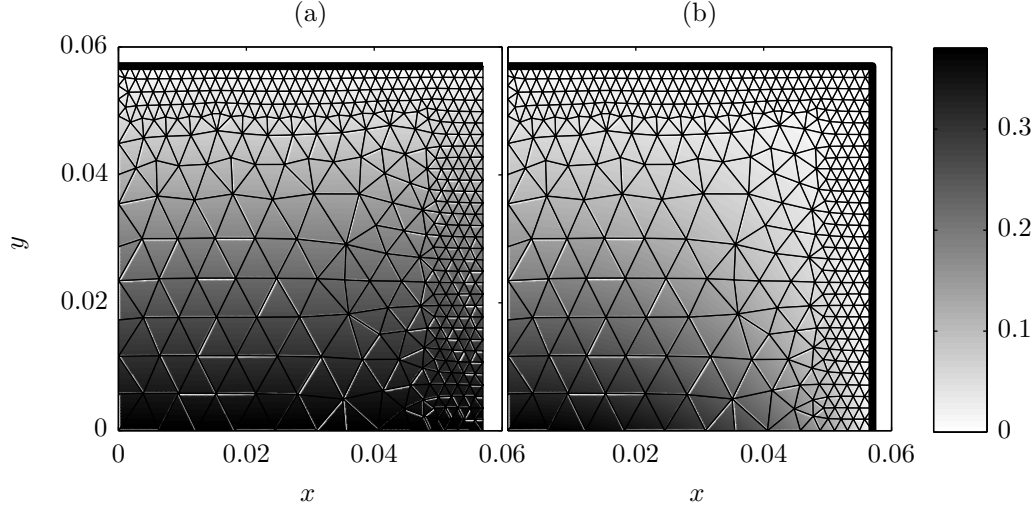


Fig. 4.7 (a) Hydrostatic pressure profile which inhibits the movement of the water column resulting from setting the boundary conditions of the pressures: only pressure nodes on the horizontal face (thick line) are set to zero which means that the flow of the water step is blocked. (b) Every node on the surface (thick line) is set to zero which is the “correct” pressure profile in the moment where the wall is removed and the water step is allowed to flow towards the right.

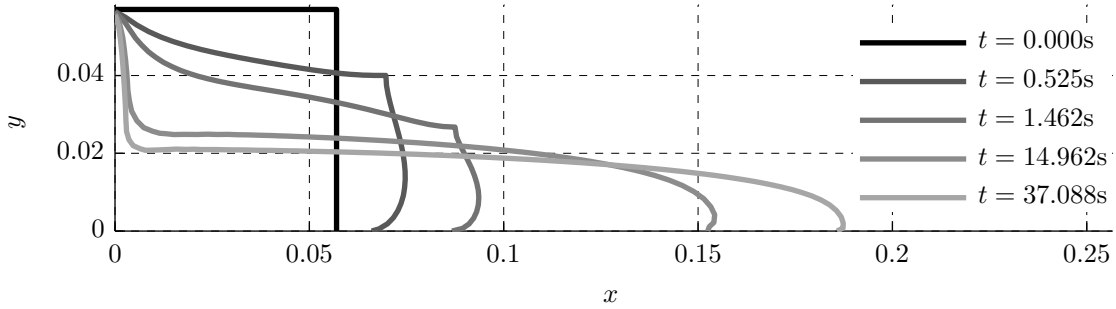


Fig. 4.8 Time evolution for the fronts in for the collapse of the square water column at $\nu = 10^{-1} [\text{m}^2/\text{s}]$. The corner travels towards the right.

for an initial area q , gravity g and viscosity ν . In the lubrication approximation, effects other than the viscosity, i.e. effects due to inertia, surface tension etc. are being neglected and the height h of the water column is much smaller than its width l [50]. Though the latter assumption is rather problematic for our case, our data in Fig. 4.9 compare rather well with Eq. (4.5) at $\nu = 10^{-1} [\text{m}^2/\text{s}]$. Changes of ν over a reasonable range of parameters did not lead to any changes in the curve. One could interpret the good correspondence as either a confirmation of our simulation method by the lubrication approximation or vice versa. The lubrication approximation predicts a faster advance than the one found in the simulation (Fig. 4.9): This is consistent with the fact that the lubrication approximation neglects inertia effects, which the simulation takes into account, and which delay the propagation of the

front. In Fig. 4.8 the shape of the corner is preserved as a nook and travels towards the right: This is an effect of our purely hydrodynamic simulation: When we tentatively included surface tension, the nook vanished.

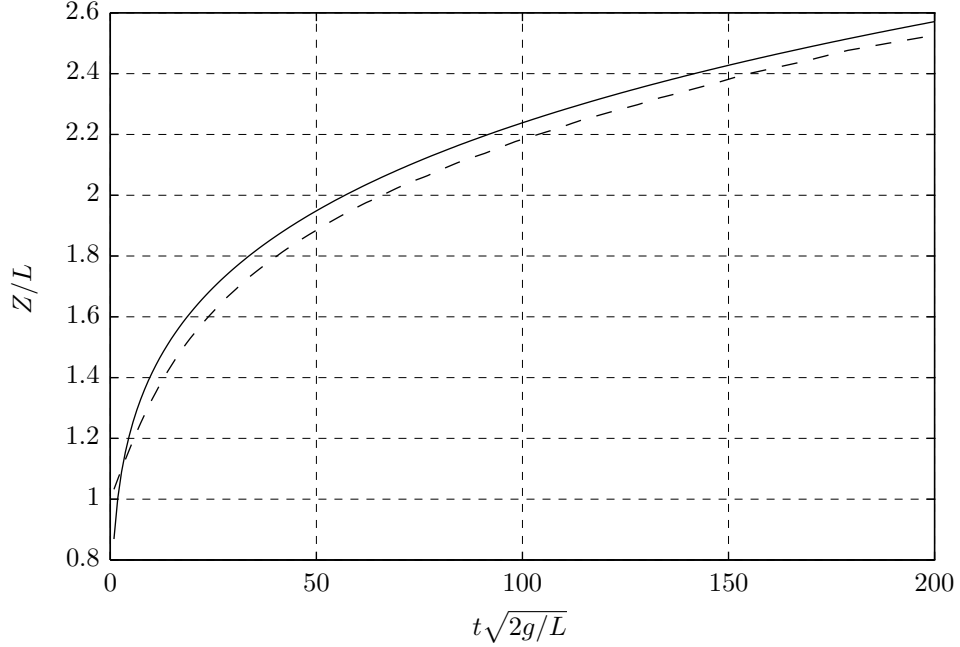


Fig. 4.9 Comparison of the time evolution of the advancing wave front under the lubrication approximation (— —) with our simulation $\nu = 10^{-1}$ [m²/s] (—).

4.4.2 Water Column with Low Viscosity

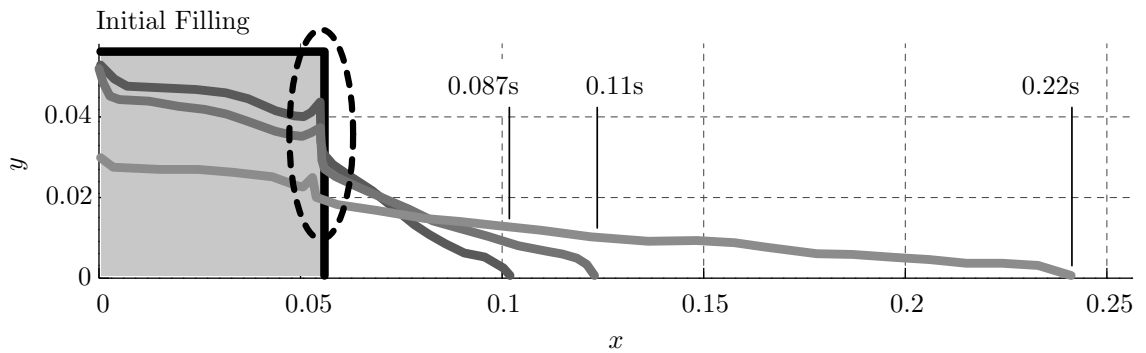


Fig. 4.10 Advance of the fluid front according to the experiment of Martin and Moyce [1] for the collapse of a fluid column of 57×57 [mm] at $\nu = 10^{-6}$ [m²/s]. The line-width is approximately the width of the shadows in the original frames, graphics is flipped compared to Martin and Moyce, for easier comparison with the simulation.

For the “breaking dam problem” at higher Reynolds numbers, no reliable analytical results exist. Therefore, we have to compare our simulation with

experimental data rather than with other simulations or theories. Especially the experiment by Martin and Moyce [1] is of interest, as for their dimensions of the vessels, air resistance can be neglected, exactly the condition which we have in our simulation. We have extracted the outlines of the evolution of the step (slightly smoothed, as the surface is denoted by shadows of strongly varying width due to scattering by the menisci on the walls in the rather dark snapshots where the frames are taken in the 1950's with a high-speed camera of 300 frames per second) and associated them with the corresponding points in time in Fig. 4.10. In the following, we focus on the shape for short time scales, which we consider more meaningful for the verisimilitude of the simulation than the long-term time evolution. For long times, geometric details are smeared out and momentum-conservation and energy-decay dominate which is why it is the preferred regime for particle methods [8].

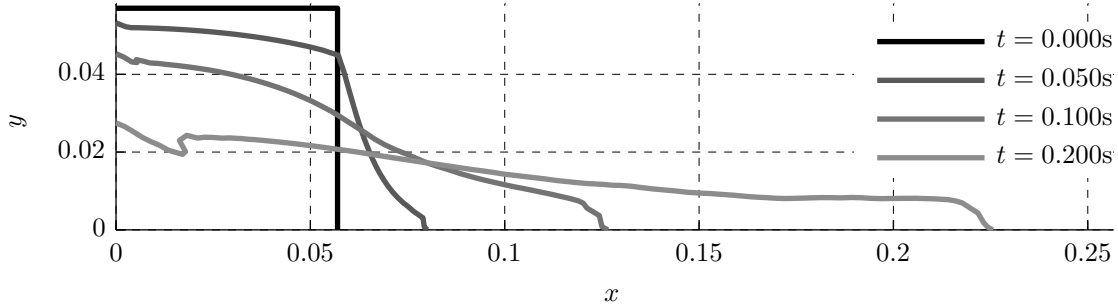


Fig. 4.11 Time evolution for the fronts in for the collapse of the square water column at $\nu = 10^{-6} [\text{m}^2/\text{s}]$ and step-size $\Delta t = 1.0 \times 10^{-5}$. The corner travels downward as in Fig. 4.10.

In the experiment Fig. 4.10, waves are forming, the surface loses convexity both on the left and on the right side. This may be due to the fact that the “release” of the dam is experimentally not unproblematic, as Martin and Moyce fixed impregnated paper with wax and the release was accomplished by melting the wax instantaneously by the current flow from an array of batteries. It is this dynamics which may trigger the formation of waves. What is striking in the high-speed pictures of Martin and Moyce [1] is, that the initial sharp step-shape between initial water level and vertical boundary (dotted oval in Fig. 4.10) is rather well preserved during the collapse, at least within the limits set by the shadows due to light scattering by the fluid meniscus. Moreover, the upper surface loses convexity both on the left and on the right side. For high viscosity $\nu = 10^{-1} [\text{m}^2/\text{s}]$ in Fig. 4.8 in Section 4.4.1, the cusp from the initial square profile had traveled fast towards the front. In Fig. 4.11 and the enlarged portion of the columns in Fig. 4.12, for the flow with low viscosity ($\nu = 10^{-6} [\text{m}^2/\text{s}]$) one can see that the upper right corner of the column travels downward as observed in the experimental data in Fig. 4.10. The resilience of the “corner” on

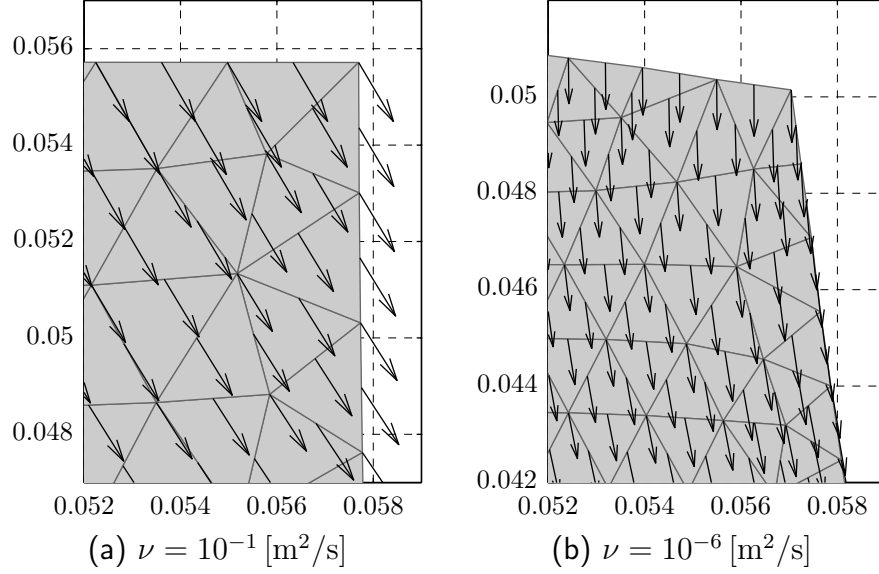


Fig. 4.12 Comparison of flow patterns at the upper right corner of the column for high (a) and low (b) viscosity at $t = 0.0375 \text{ [s]}$. For high viscosity fluid, the upper right corner travels towards the front while for low viscosity fluid, the corner travels downward.

the right is rather surprising, not to say counter-intuitive, but as the pressures in upper right corner are negligible, there are no forces which could cause the decay of the angle. The different behavior of the corner can be understood when we look at the flow field in the simulation in Fig. 4.11: While for high viscosity fluid, the flow field of the upper right corner points diagonally downward out of the fluid front, for low viscosity the flow field points downward parallel to the front. On the left side of the simulation in Fig. 4.11, a nook-shaped surface instability develops between $t = 0.1$ and 0.2 [s] which would be suppressed if surface tension were implemented: The fluid level sinks faster than the meniscus on the left boundary. The height gradient and the flow away from the boundary introduces a “downhill-flow” towards the right, which bumps into the slower moving surface and flushes up a wake, as in water rapids.

While the shape of the wave-front for low viscosity flow may look unphysical,

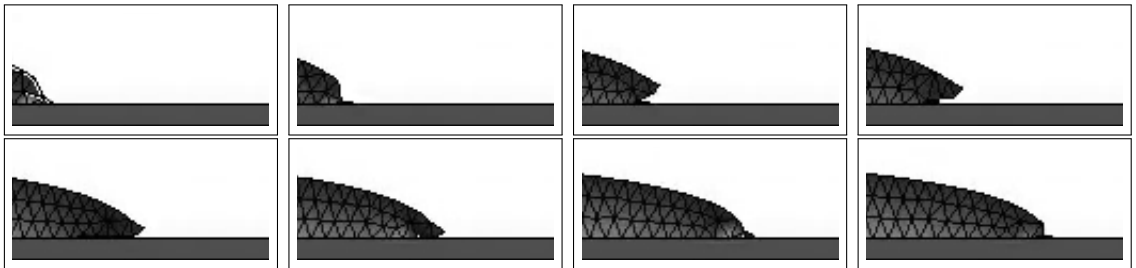


Fig. 4.13 Snapshots showing the shape of the wave-front for low viscosity flow.

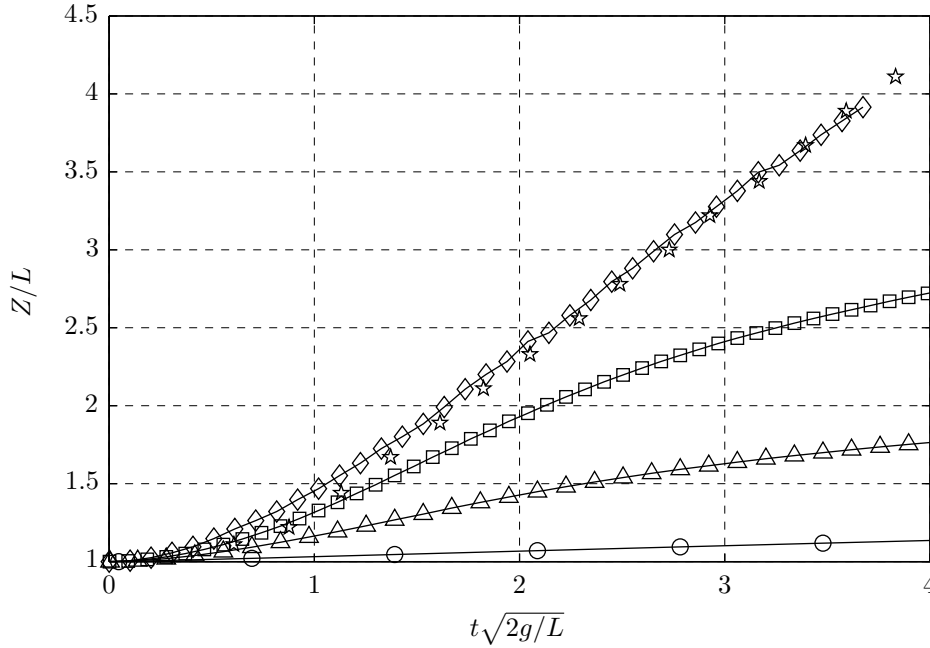


Fig. 4.14 Comparison of the computed time evolution of the advance of the fronts with viscosity of $\nu = 10^{-1}$ (\circ), $\nu = 10^{-2}$ (\triangle), $\nu = 10^{-3}$ (\square), and $\nu = 10^{-6}$ (\diamond), with the experimental data of Martin and Moyce [1] (\star).

the intention of this study was to find out how accurately the flow velocity could be reproduced with large grid size for a purely “hydrodynamical simulation” without the implementation of surface tension; For physical systems, adhesion with the surface and surface tension will modify the shape, but not the flow velocity. Further, the stability of the simulation for such relatively large mesh sizes proof that “stability” is often not an issue for the FEM-simulation.

In Fig. 4.14 we show the time evolution of the advance of the front of water columns with different viscosity with the experimental data from Martin and Moyce [1]. For low viscosities, i.e. that of water, we were able to obtain a very good agreement with the experimental data (diamonds in Fig. 4.14). That the speed of the wavefront from the experiment is reproduced correctly by the simulation is also a corroboration of the validity of the FEM-approach, as the surface modeling with the AB-integrator works with the velocities from the FEM-approach without needing the additional data structures as the conventional ones.

Chapter 5

Simulation of Fluid with Suspended Particles

This chapter shows how we combine the simulation for the granular particles with the simulation for the fluid. This includes the choice of boundary conditions of the fluid, and the computation of external forces on the particles. The particle-fluid simulation is then being verified via the computation of the wall correction factor.

5.1 Coupling the Discrete Element Method and the Finite Element Method

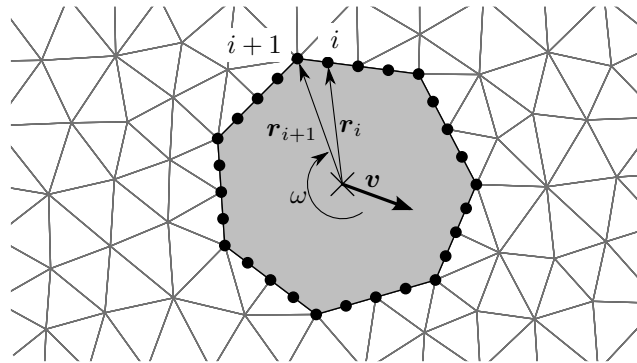


Fig. 5.1 Non-slip boundary conditions on the nodes \bullet around the particles: Velocities $\mathbf{v}_i^{\text{bc}} = \mathbf{v} + \omega \times \mathbf{r}_i$ on the i -th node are computed from the predicted velocity \mathbf{v} and angular velocity ω of the particle.

We have shown in Chapter 2 and Chapter 3 that we have working formulation of both particle and fluid simulation. For the fluid to “see” the particle, the boundary condition of the interfaces are set to non-slip, i.e. the flow velocities of the fluid

boundary around the particle are computed from the predicted velocity and angular velocity of the particle (see Fig. 5.1) as the velocities of the respective point on the particle boundary. Fig. 5.2 shows the flowchart of the simulation when combining the particle simulation with the fluid simulation. The box with gray background indicates that the process belongs to the discrete element method in Chapter 2.

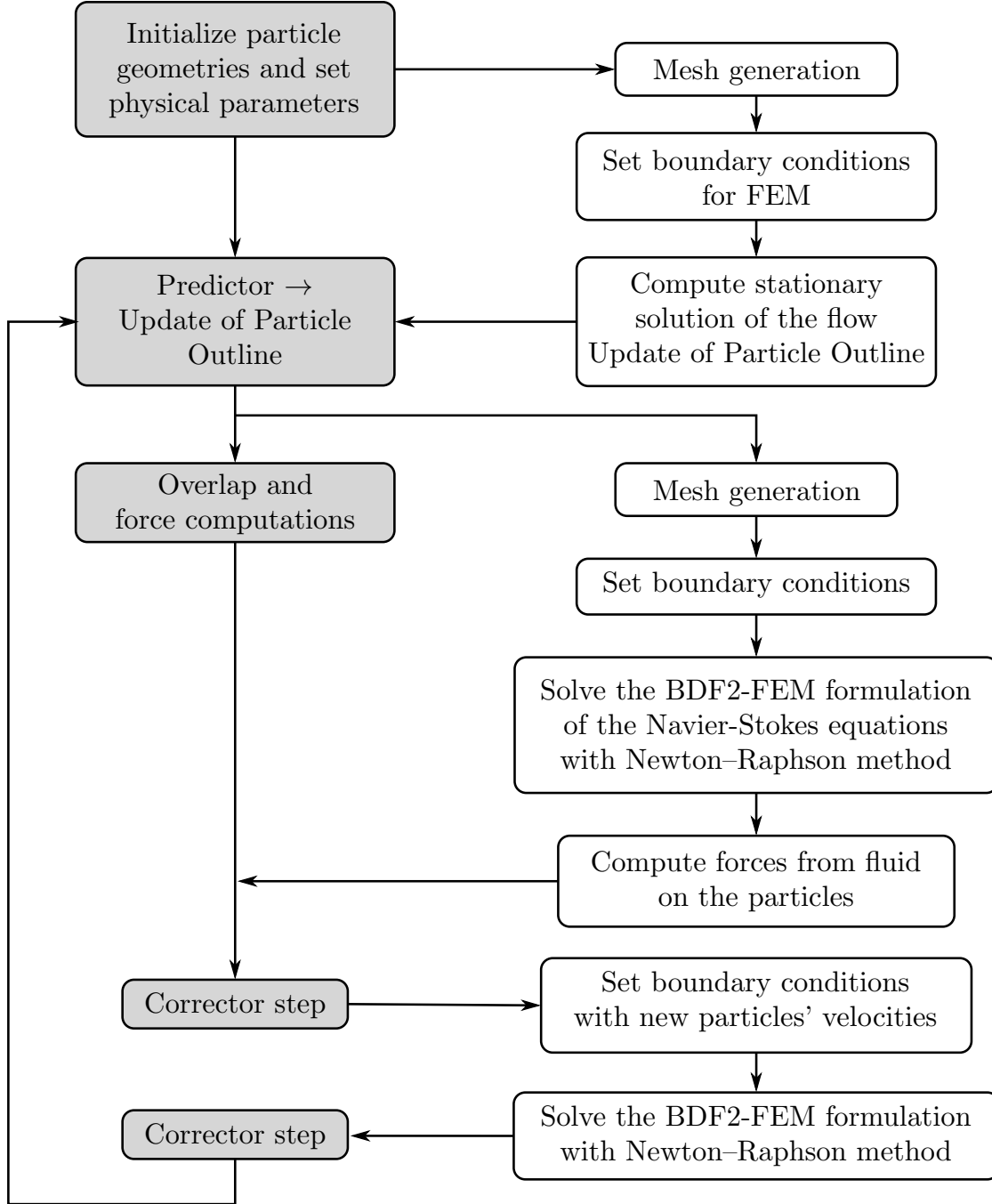


Fig. 5.2 Flowchart of the simulation fluid with suspended polygonal particles. Processes which belong to the discrete element method are on the left (in gray).

After solving the BDF2-FEM formulation of the Navier-Stokes equation, the

force from the flow which acts on the particle is computed by integrating the fluid stress tensor over the particle's surface [51]

$$\mathbf{F}^D = \int_{\Gamma} \{-p\delta_{ij} + \mu_f((\nabla \mathbf{u}) + (\nabla \mathbf{u})^T)\} \cdot \hat{n} \, dl, \quad (5.1)$$

where the first part describes the normal forces on the boundary due to the pressure, and the second part the tangential forces due to the drag. The δ_{ij} is the usual Kronecker delta with

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j, \end{cases} \quad (5.2)$$

μ_f [Pa · s] is the dynamic viscosity, and the Jacobian matrix $\nabla \mathbf{u}$ denotes the velocity gradient

$$(\nabla \mathbf{u})_{ji} = \frac{\partial u_j}{\partial x_i}. \quad (5.3)$$

\hat{n} denotes the normal vector of the surface element pointing outwards from the particle (see Fig. 5.4). Eq. (5.1) is comprised of a pressure part (form drag)

$$F_x^{Dp} = \int_{\Gamma} -p \cdot n_x \, dl, \quad (5.4)$$

$$F_y^{Dp} = \int_{\Gamma} -p \cdot n_y \, dl, \quad (5.5)$$

which becomes finite when the particle moves different velocities than the fluid in the propagation direction. The viscous part (friction drag)

$$F_x^{Dv} = \mu_f \int_{\Gamma} 2n_x \frac{\partial u}{\partial x} + n_y \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \, dl, \quad (5.6)$$

$$F_y^{Dv} = \mu_f \int_{\Gamma} 2n_y \frac{\partial v}{\partial y} + n_x \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \, dl \quad (5.7)$$

is finite if the fluid velocity is different from the particle velocity to the left and right in propagation direction and a velocity gradient towards the particle boundary is finite. The shape of the object and the angle of attack determine which type of drag will have higher effect on the object. If the object has a streamlined body, the drag force will be dominated by the viscous part; whereas if the object is a bluff body, the force will be dominated by the pressure part (see Fig. 5.3). To put the computation of the force in practice, in our code:

- We look for all FEM-elements whose have an edge which is on the border of a particle boundary. (Elements which have only a corner on a particle boundary

do not have to be dealt with, as these corners are also corners of particles with edges on the particle boundary.)

- For each element, we compute the form drag and friction drag using the numerical equivalents for for Eq. (5.1), respectively Eq. (5.4)–(5.7) which will be discussed in the following sections.
- The resulting force is then obtained by summing all the fluid forces which result from elements on the particle boundary. Additionally, the external forces have to be computed.
- Finally, from the forces on the boundary the torques on the particle are evaluated.

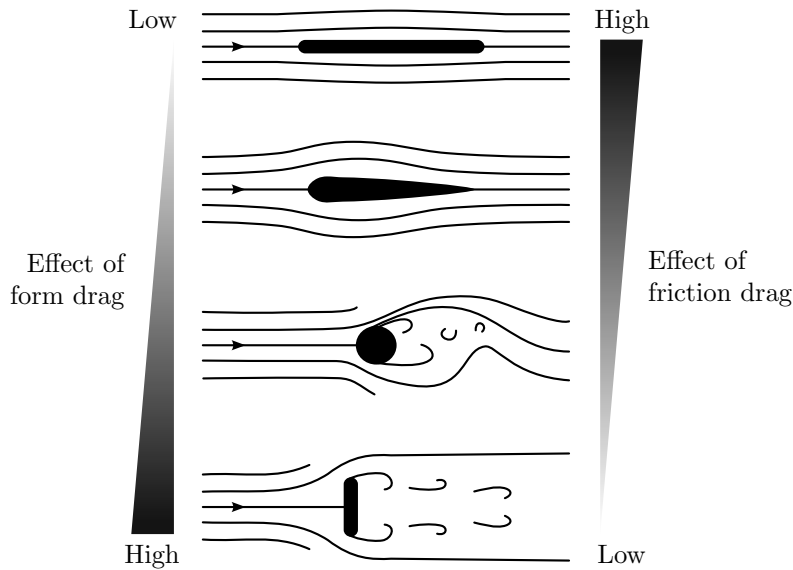


Fig. 5.3 The effect of the form drag and friction drag on the object immersed in fluid depends on the shape and angle of attack: streamlined body (top) will be dominated by the friction drag, bluff body (bottom) will be dominated by the form drag (modified according to Nakayama et al. [52].)

5.1.1 Form Drag

The forces can be obtained from the elements which are attached on the particle's edge (see Fig. 5.4). Recall that in Chapter 3 we explained that the approximation of pressures with P_2P_1 -elements uses affine functions as

$$p = \psi^\top \mathbf{p},$$

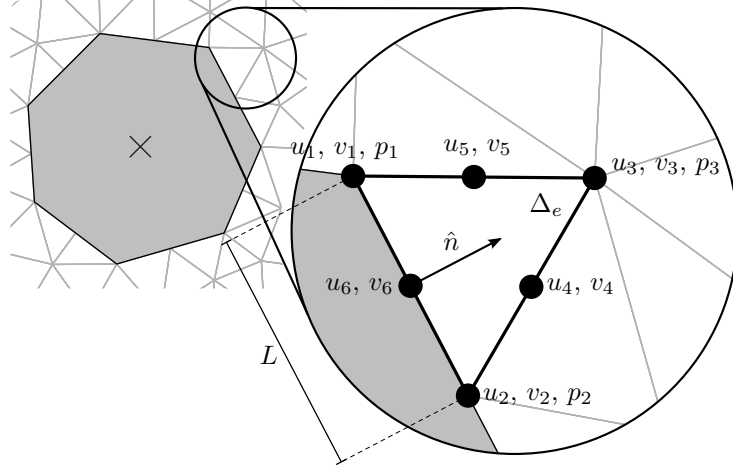


Fig. 5.4 Computation of the fluid force on the particle from the nodal values (u_i, v_i, p_i) of an FEM-element Δ_e . \hat{n} denotes the normal vector of the surface element. As only the forces on lines of finite length are relevant, elements which have only a corner on the boundary of the particle do not have to be treated.

and the shape functions ψ can be written in the local area coordinates $\psi = \{L_1 \ L_2 \ L_3\}^\top$ so that we have the pressures as

$$p = L_1 p_1 + L_2 p_2 + L_3 p_3. \quad (5.8)$$

To obtain the form drag in x -direction, we substitute the above into Eq. (5.4)

$$F_x^{\text{Dp}} = -n_x \int_{\Gamma} (L_1 p_1 + L_2 p_2 + L_3 p_3) \, dl. \quad (5.9)$$

For the element Δ_e in Fig. 5.4, since the integration will be performed along the edge from the first to the second node on the particle's boundary, L_3 will be zero along the edge, so the drag force becomes

$$(F_x^{\text{Dp}})_{\Delta_e} = -n_x \int_{\Gamma} (L_1 p_1 + L_2 p_2) \, dl, \quad (5.10)$$

Integrating over the one-dimensional line element Eq. (3.58) yields

$$(F_x^{\text{Dp}})_{\Delta_e} = -n_x \frac{p_1 + p_2}{2} L. \quad (5.11)$$

which is the average of the nodal value of the pressures on the particle's edge, where L is the length of the edge. The y -direction form drag is obtained analogously as

$$(F_y^{\text{Dp}})_{\Delta_e} = -n_y \frac{p_1 + p_2}{2} L. \quad (5.12)$$

5.1.2 Friction Drag

The computation of the viscous part of the force is more complicated as it involves the gradients of the velocities which in the P₂P₁-elements are approximated with quadratic functions. We start with the computation of the partial derivative of u with respect of x using the local coordinates $\mathbf{L} = \{L_1 \ L_2 \ L_3\}^\top$,

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{\partial u}{\partial L_1} \frac{\partial L_1}{\partial x} + \frac{\partial u}{\partial L_2} \frac{\partial L_2}{\partial x} + \frac{\partial u}{\partial L_3} \frac{\partial L_3}{\partial x}, \\ &= \frac{\partial \mathbf{L}^\top}{\partial x} \frac{\partial u}{\partial \mathbf{L}}. \end{aligned} \quad (5.13)$$

The velocity u is written again as the linear combination of the basis functions

$$u = \boldsymbol{\varphi}^\top \mathbf{u}, \quad (5.14)$$

where the $\boldsymbol{\varphi}$ are formulated in local area coordinates as

$$\boldsymbol{\varphi} = \left\{ L_1(2L_1 - 1) \quad L_2(2L_2 - 1) \quad L_3(2L_3 - 1) \quad 4L_2L_3 \quad 4L_3L_1 \quad 4L_1L_2 \right\}^\top. \quad (5.15)$$

Here, $\mathbf{u} = \{u_1 \ u_2 \ \dots \ u_6\}^\top$ are the nodal values of the velocities obtained as the solution of the Navier–Stokes equations. Substituting the velocities into Eq. (5.13) yields

$$\frac{\partial u}{\partial x} = \frac{\partial \mathbf{L}^\top}{\partial x} \frac{\partial \boldsymbol{\varphi}^\top}{\partial \mathbf{L}} \mathbf{u} \quad (5.16)$$

$$= \frac{\partial \mathbf{L}^\top}{\partial x}, \begin{bmatrix} 4L_1 - 1 & 0 & 0 & 0 & 4L_3 & 4L_2 \\ 0 & 4L_2 - 1 & 0 & 4L_3 & 0 & 4L_1 \\ 0 & 0 & 4L_3 - 1 & 4L_2 & 4L_1 & 0 \end{bmatrix} \mathbf{u}. \quad (5.17)$$

Referring back to Eq. (3.55) and Eq. (3.56) in Section 3.2.4, we know that both $\frac{\partial \mathbf{L}}{\partial x}$ and $\frac{\partial \mathbf{L}}{\partial y}$ are independent from the local coordinates

$$\frac{\partial \mathbf{L}}{\partial x} = \left\{ B_1 \quad B_2 \quad B_3 \right\}^\top, \quad (5.18)$$

$$\frac{\partial \mathbf{L}}{\partial y} = \left\{ C_1 \quad C_2 \quad C_3 \right\}^\top, \quad (5.19)$$

and can be computed from positions of the vertices of the triangular mesh using Eq. (3.46) and Eq. (3.47). Now it is clear that to evaluate the integral on the velocity

gradients Eq. (5.1), we only have to evaluate the matrix $\frac{\partial \boldsymbol{\varphi}^\top}{\partial \mathbf{L}}$

$$\int_{\Gamma} \frac{\partial u}{\partial x} dl = \frac{\partial \mathbf{L}^\top}{\partial x} \left[\int_{\Gamma} \frac{\partial \boldsymbol{\varphi}^\top}{\partial \mathbf{L}} dl \right] \mathbf{u}, \quad (5.20)$$

using Eq. (3.58). Again with Fig. 5.4 as an example where $L_3 = 0$ along the particle's edge, we obtain

$$\begin{aligned} \int_{\Gamma, 1 \rightarrow 2} \frac{\partial \boldsymbol{\varphi}^\top}{\partial \mathbf{L}} dl &= \int_{\Gamma} \begin{bmatrix} 4L_1 - 1 & 0 & 0 & 0 & 0 & 4L_2 \\ 0 & 4L_2 - 1 & 0 & 0 & 0 & 4L_1 \\ 0 & 0 & -1 & 4L_2 & 4L_1 & 0 \end{bmatrix} dl \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & -1 & 2 & 2 & 0 \end{bmatrix} L. \end{aligned} \quad (5.21)$$

When the connecting edge is formed by the second and third nodes ($L_1 = 0$),

$$\int_{\Gamma, 2 \rightarrow 3} \frac{\partial \boldsymbol{\varphi}^\top}{\partial \mathbf{L}} dl = \begin{bmatrix} -1 & 0 & 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \end{bmatrix} L, \quad (5.22)$$

and if the edge is formed by the first and third nodes ($L_2 = 0$),

$$\int_{\Gamma, 1 \rightarrow 3} \frac{\partial \boldsymbol{\varphi}^\top}{\partial \mathbf{L}} dl = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 & 0 \\ 0 & -1 & 0 & 2 & 0 & 2 \\ 0 & 0 & 1 & 0 & 2 & 0 \end{bmatrix} L. \quad (5.23)$$

Using the above approach, we can compute the friction drag from an triangular element which is attached on the particle's boundary.

5.1.3 Modeling a Shadow around the Particle

In three dimensional granular materials, one region of the pore-space is practically never closed off from another region, there are practically always channels between the particles. In two dimensions, as in Figure 5 a), by default it is easy that already three particles form a region of the pore-space which is separated from the remaining pore-space. To model the pore-space in such a way that blocking of pore regions is avoided also in two dimensions as in the realistic three dimensional system, the particles are modeled with a core, whose boundary is a boundary of the fluid, and a "shadow" which is used to for the computation of the interaction between the

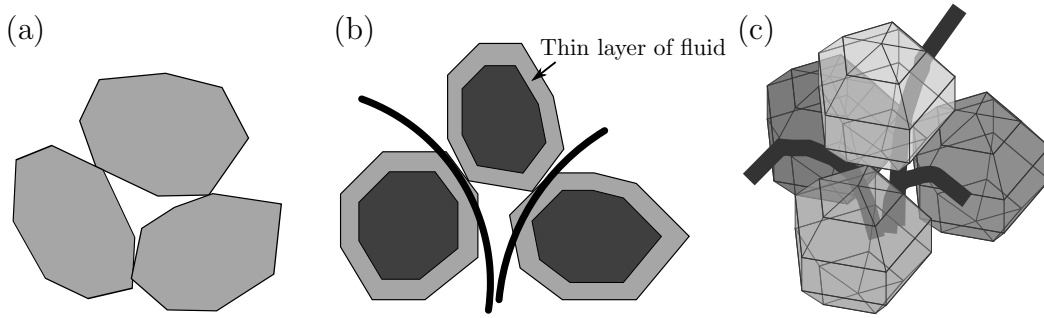


Fig. 5.5 Separated space in straightforward modeling of two-dimensional particles without shadow in (a), (b) Fluid space (white), shadow by which the particles interact (light gray) and core (dark gray) which forms the boundary of the fluid flow and (c) Three-dimensional arrangement of grains which leads to flow between the particles along the thick lines which is supposed to be mimicked by the shadow on the left.

granular particles. Inside this shadow, the fluid flow is computed as for the empty space to obtain a connected pore space, so that no sub-volumes are closed off, as would be the case for three dimensional particles, see Fig. 5.5. The size ratio between the whole particle and the core is about $1 : 0.72$. The physical meaning of the pore space is that it is formed by particle asperities of the rods which are equivalent to the two-dimensional particles so that a connected pore space is formed (see Fig. 5.6).

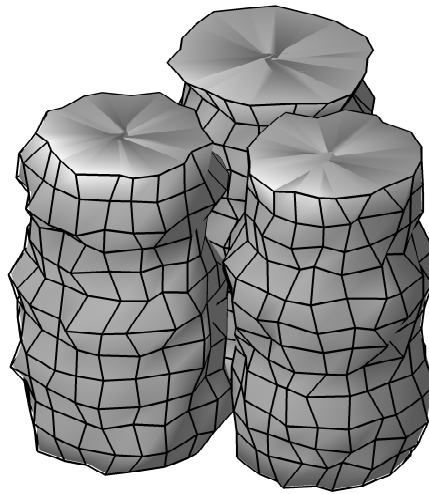


Fig. 5.6 Pore space formed by particle asperities of the rods which are equivalent to the two-dimensional particles.

5.2 Stability and consistence: Two Square Particles in Fluid

In this section, we will investigate various effects on the simulation under “adverse” condition, to see which influences would destabilize the simulation so much that it would become infeasible. First, the physics in the simulation should be independent of the structure of the underlying grid. Before we tested for correctness, we at least wanted to see whether there are any artifacts which would even prevent consistency.

A simple test was conducted by releasing two particles (which were separated with equal distance from the left and right wall, respectively) from the height of 10.0 [mm] inside a domain of size 60.0×30.0 [mm]. The difference of dropping velocities in this symmetrical configuration would indicate possible problems due to lack of symmetry, stability etc.. Square particles (with length 4.24 [mm]) were chosen as the right angles at the corners will give the worst flow singularities imaginable for convex particles. After the particles are released, they sink under the influence of gravity and come in contact with the ground which is also treated as a particle in the discrete element method. The dynamic viscosity of the fluid is set to 10^{-3} [Pa · s] which gives a Reynolds number of 9.8×10^{-5} . The density for both particles and the wall is 10^4 [kg/m³] while the density of the fluid is 1 [kg/m³]. Both discrete element method and finite element method use the same fixed time-step $\tau = 7.5 \times 10^{-5}$ [s]. This means that 15 time-steps are needed to resolve a single dry collision, i.e. a collision between particles which are not slowed down by the surrounding fluid and which have no cohesive interaction.

In Fig. 5.7, we show the snapshots of the pressure distribution and the velocity field of the fluid around the particles before and after their collision with the ground. In order to observe the effect of the fluid on the particles, we compare with the same configuration without the fluid. In the plot of the center of mass (in y -direction) of the particle on the left for both simulations in Fig. 5.8, we can see that when the particles are sinking, the pressure below them increases. The increase of the pressure and velocity gradient in the fluid results in a larger force which decelerates the particles. Therefore, in fluid the velocity of the particle is decreasing before the collision while in the dry case the velocity of the particle is maximal during the collision. That the amplitude of the form drag is larger than the friction drag indicates that we are in the regime which is dominated by the pressures as shown in Fig. 5.3. Due to the damping effect from the fluid, the particles take longer time to reach the bottom compared to the situation without fluid. We can also see that

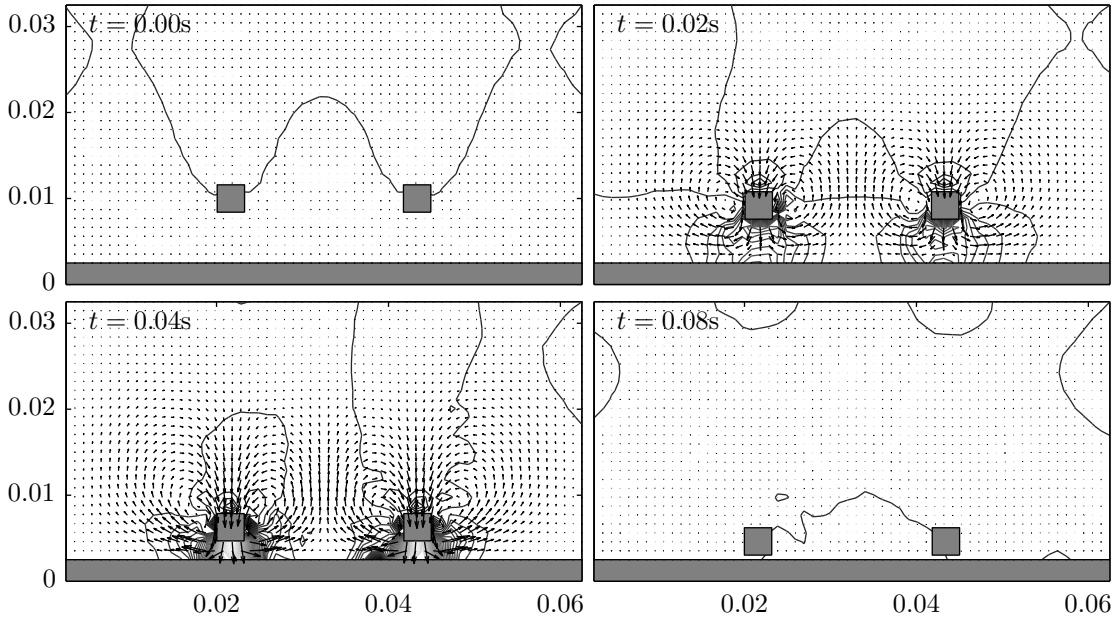


Fig. 5.7 Snapshots with velocity fields and pressure distributions of two square particles sinking under the influence of gravity after being released in a symmetrical configuration.

the particle bounces upward after the collision for the dry case but not in the fluid which additionally damps the kinetic energy.

The spikes in the form drag are due to the sudden shape changes of the finite element grid in the constrained Delaunay decomposition during the simulation. The time integration with these forces (positions and velocities) is nevertheless stable. This shows that the stiffly stable integrator the BDF2 fulfilled its purpose: Forces with non-smooth time-evolutions can be integrated up to smooth velocities and positions for the particles. In this examples, the triangular meshes were chosen so large that the edge length were comparable to the edge length of the particle (is that true, then it should also be mentioned further above). This results in significant force changes during remeshing. While these force changes could be reduced by choosing a smaller grid size, in this example we wanted to see whether the limit (minimal degrees of freedom for the mesh, therefore maximal speed for maximal size of the triangles) would still give a feasible simulation of the particles (smooth velocities and trajectories). In our approach, the effect on the particles is more important, as the particle trajectories can more easily be measured experimentally than the flow field between neighboring particles, so we would be willing to work with unphysical flow fields as long as the particle flow is correct for certain cases.

Such stability is not a matter of course: In the next step, we wanted to test how our simulation was affected by the order of the integrator for the particles. When we

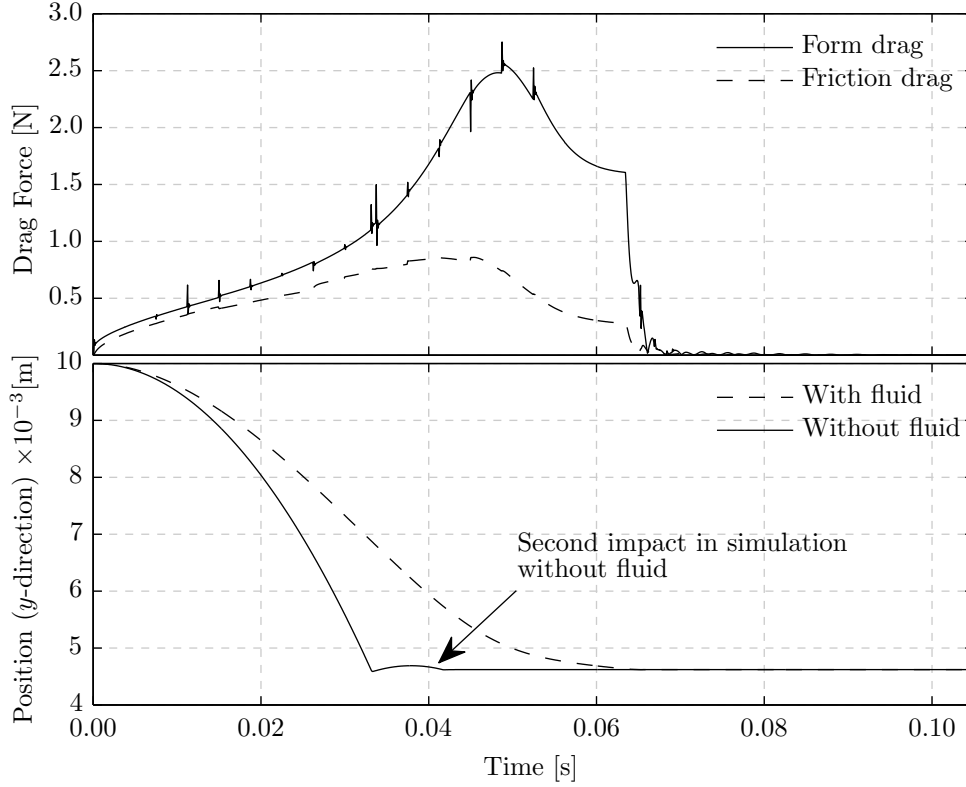


Fig. 5.8 Above: Form drag and friction drag from the fluid on the left particle. Below: Center of mass (in y direction) of the left particle for both simulation with and without fluid.

replaced the Gear predictor-corrector of second order (BDF2) with the fifth order (BDF5) for the particles the combined fluid-particle code either “explodes”, (the forces oscillate with increasing magnitude), or residual oscillations with a period of a single time-step remain if the force from the fluid is reduced by over an order of magnitude. This is rather surprising, as the higher order BDF5 is constructed so that the stability should be rather better than BDF2, and for the non-smooth interactions of the DEM-particles we never encountered any problems. As the BDF5 integrator uses the information of five time-steps, BDF2 uses the information of only two time-steps: The delayed arrival of the forces from the particles creates noise for the flow field which cannot be damped out by the fluid part, and feeds back into the particles. We have to draw the conclusion that time integrators which by themselves as stable for fluid or particle simulations can be incompatible (even if they are of the same family) and lead to instabilities if two in itself stable simulations are combined. Further destabilization can occur if the mesh is restructured too much between the predictor and the corrector step: for large meshes, the change of the flow from one mesh to another is not smooth, which in itself does not necessarily lead to a breakdown of the simulation.

5.2.1 Verification with Wall Correction Factor

While drag coefficients are the conventional quantity which are used to characterize the fluid resistance for bodies in infinite regions, for systems with many particles the wall correction factor (how much larger is the drag in a narrow geometry compared to the infinite system) is a more meaningful parameter. Although the particles in our simulation are convex polygons, we compare the value of the wall correction factor $\lambda(k)$ with Richou et al. [53] for a circular cylinder due to lack of comparable reference data for polygonal shape. For this, we chose a dodecagon (12 corners) as an approximation to the circular shape, to see how close we come to the values for circular particles. We selected this force computation for sinking particles between walls because it allows conclusions concerning the accuracy of the particle simulation and fluid simulation together with the treatment of the boundaries.

The system is set up so that a circular cylinder with radius r is released in a channel of width $2b$ as shown in Fig. 5.9. The wall correction factor $\lambda(k)$ of the drag force is given as

$$\lambda(k) = \frac{F_y(k)}{\mu \cdot U_0}, \quad (5.24)$$

where $F_Y(k)$ is the vertical drag force, μ is the dynamic viscosity of the fluid, U_0 is the terminal (sinking-) velocity and k is the aspect ratio between the radius r and the half of the channel width b

$$k = \frac{r}{b}. \quad (5.25)$$

The two parallel walls are of height 0.75 [m], while the particle is released at a

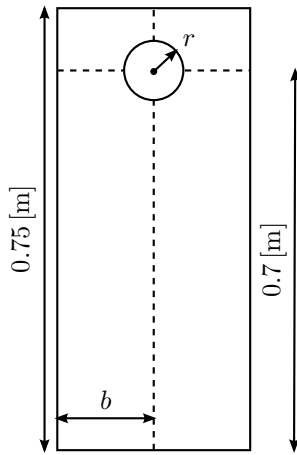


Fig. 5.9 Dimensions of the system for computing the wall correction factor. The width of the system is determined by Eq. (5.25).

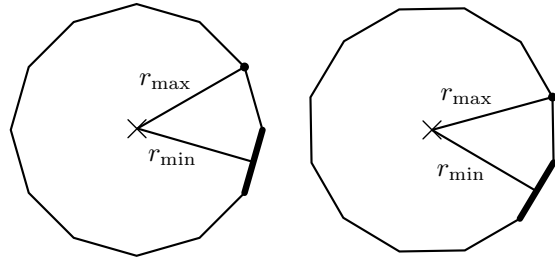


Fig. 5.10 Initial orientations of the polygon for computing the wall correction factor with corner-down (left) and edge-down (right).

height of 0.7 [m]. Since the cross-section of our cylinder is a polygon instead of a perfect circular disk, there is a certain arbitrariness in the choice of the “radius” of the corresponding approximated disk. While the distance between the center to a corner was $r_{\max} = 0.015$ [m], alternatively we could also use the the closest distance to the boundary

$$r_{\min} = r_{\max} \cdot \cos(\pi/n), \quad (5.26)$$

where $n = 12$ is the number of corners of the polygon (see Fig. 5.10), or the radius of the disk with the same area A ,

$$r_{\text{aver}} = \sqrt{A/\pi}. \quad (5.27)$$

The aspect ratio of k is fixed to 0.125 which gives wall correction factor $\lambda(k) = 10.8020$ for circular particles at $Re = 2 \times 10^{-4}$ [53]. For our dodecagon, the system width b will be

$$b_{\max} = r_{\max}/0.125, \quad (5.28)$$

$$b_{\min} = r_{\min}/0.125, \quad (5.29)$$

$$b_{\text{aver}} = r_{\text{aver}}/0.125, \quad (5.30)$$

respectively. Computations are both done for particle for “corner-down” and “edge-down” orientations (see Fig. 5.10). The density of the fluid is 1000 [kg/m³] while the density of the particle is 5000 [kg/m³]. The viscosity of the fluid is chosen as 500 [Pa · s] so that we obtain Reynolds number comparable to $Re = 2 \times 10^{-4}$ in Richou et al. [53].

In Table 5.1, we compare our numerical results with Richou et al. for the aspect ratio $k = 0.1250$. As we are far enough away from the ground (the measurements are taken in the upper third of the vessel), our values are not corrected with respect to the influence from the ground, and our results are close to Richou et al. with deviation of the order of 0.2% to 6%. The value from Richou et al. deviates from the theoretical values from the formulae by Faxen and Takaishi (cited after Richou) by up to 0.23%. If we assume that the most important contribution is the distance from the particle to the wall, the value for r_{\max} for the edge-down particle is the closest to that of disks, and the agreement can be considered excellent.

It is interesting to compare how other approaches fare in the computation of this problem. The limiting value for the terminal velocity for a particle falling in a fluid was computed by Ristow [2] via a MAC-scheme. While the MAC-grid itself was square, the particle was assumed to be circular, so that the varying overlap

Table 5.1 Comparison between our numerical and Richou et al.'s results of wall correction factor $\lambda(k)$ for $k = 0.1250$.

		Corner-down			Edge-down		
		r_{\max}	r_{\min}	r_{aver}	r_{\max}	r_{\min}	r_{aver}
$n = 12$	U_0 [mm/s]	5.89	5.74	5.86	6.12	5.94	5.97
	$\lambda(k)$	11.24	11.54	11.29	10.83	11.14	11.09
$\lambda(k)$, Richou et al. (circular disk)				10.8020			
$\lambda(k)$, Faxen (1946). (circular disk)				10.5574			
$\lambda(k)$, Takaishi (1955). (circular disk)				10.7958			

between particle and meshes was implemented by the boundary conditions of on the particle: Choosing the velocity as no-slip (zero) led to velocities with a negative sign inside the mesh-grids closest to the particle wall compared to the nearest velocities outside the particles, with different magnitudes depending on the distance of the mesh points inside and outside the particle to the particle boundary. A remeshing and an exclusion of the particle volume from the fluid domain in this approach was not possible. In his study, the fluctuations (see Fig. 5.11 (right)) due to the varying overlap of the particle over the meshes varied between 2.6% and 5.6% for particle's diameter of 1 [cm] to 3 [cm] while the particles had to be resolved in each direction with the order of then grid-points, i.e. hundreds of MAC-meshes. In Fig. 5.11 (left), we show our data for a comparable channel and particle diameter of 3 [cm] and $Re \approx 35$ where the fluctuations are only 0.56%, while the particle occupies the area of about 15 FEM-triangles. Even taking the larger number of degrees of freedom for

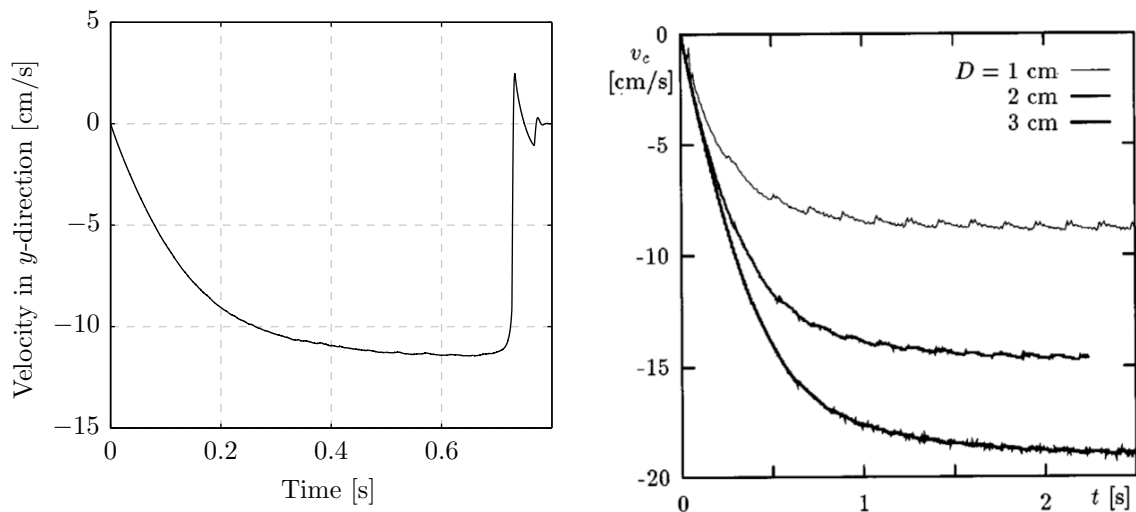


Fig. 5.11 Left: Vertical velocities for a sinking particle via our DEM-FEM code for diameters $D = 3$ [cm], $\mu = 1.0$ [Pa · s], $Re \approx 35$. Right: Terminal velocity after Ristow [2, Fig. 2] via a MAC-scheme.

the FEM into account, our approach is vindicated with respect to smoother data for the force evolution with much smaller number of the degrees of freedom. The snapshots of the same simulation are shown in Fig. 5.12.

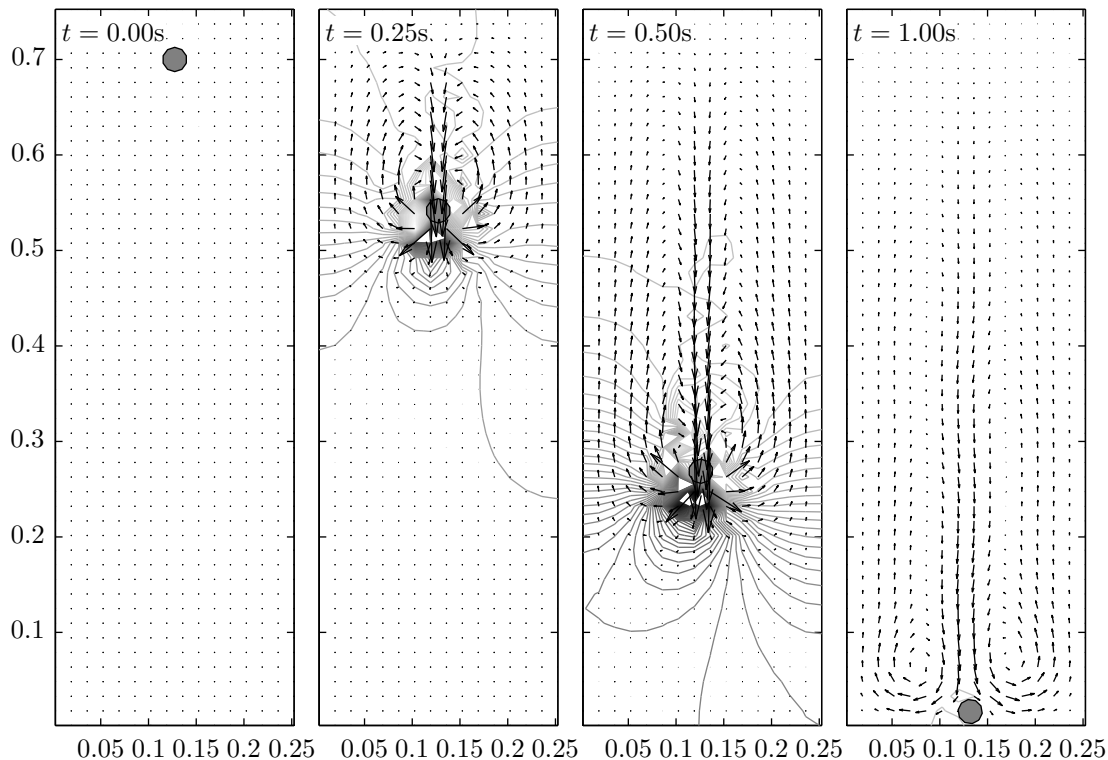


Fig. 5.12 Snapshots with velocity fields and pressure distributions of sinking a dodecagon under the influence of gravity for the computation of the terminal velocity.

We can conclude that the common approach of “macroscopic simulations” which use superficially “cheap” round particle simulations and “cheap” square grids for particle in fluid is not economic at all, as one has to deal with much smaller grids and smaller step-sizes and obtains much noisier force-fields acting on the particles nevertheless. For the future purpose of simulating fluidization phenomena in landslides and earthquakes, such large noise-amplitudes endanger the physicality and therefore the meaning of the whole simulation. With our current approach with polygonal particles in triangular grids, we see that we can obtain minimal noise amplitudes and minimal degrees of freedom while results for round particles are still reproducible.

Chapter 6

Applications of the Code

Using the approaches discussed in the previous chapters we conducted two kinds of numerical experiments: one with a rather slow dynamics, another one relatively more “violent”. The former will be the investigation of the compaction due to tapping in two-dimensional granular columns: Here the motion of the particles is not significant, the distance covered is only a fraction about 2% of the particle diameter over the whole simulation time scale. The latter is the collapse of a granular column in fluid. For both simulations, we compare the dry system with the system immersed in a viscous fluid.

6.1 Effect of the Surrounding fluid on the Compaction of Granular Materials by Tapping

Conventionally, the equilibrium state is considered to be the state which is obtained after “long enough” times so that no changes takes place any more. For many physical systems, the relaxation from the non-equilibrium state to the equilibrium proceeds exponentially. Nevertheless, near phase transitions the relaxation is according to power law behavior, and accordingly, when critical points are approached, there is a mixture between power-law and exponential behavior. In the field of “slow dynamics”, phenomena which show power-law (or slower) behavior even though the system is not near a phase transition are of particular interest. Power-law behavior allows the methodology (mathematical and otherwise) from the field of phase transitions to be applied, while even slower relaxation is associated with disorder transitions (“spin glass systems”).

Since the compaction due to tapping is logarithmically slow [54, 55, 56], the relaxation of granular materials is considered as a paradigm of the field of “slow” dynamics. For realistic granular materials, the questions of what is an equilibrium state is not easily settled as the definition of “long enough” in the presence of static (Coulomb) friction, which acts effectively as a constraint of motion, is still not clear. If a granular material which is initially in a static state is excited, depending on the excitation (shaking, vibration, pneumatic driving . . .), the resulting density may be higher or lower than that of the static state before the excitation. The compaction under tapping (acceleration of boundaries) of granular columns is here investigated computationally so that between the excitations, the system can return again to a static state for both the case of dry material, as well as for particles fully immersed under a Newtonian viscous fluid.

6.1.1 Previous Studies

Most of the experiments conducted to investigate tapping were for round glass beads [57, 58, 59, 60]. However, more irregularly shaped [61] and even needle-shaped particles [62] have been investigated recently. In the following, and for the sake of the simulation, we will assume that the propagation of the “shock” will be through the granular material only, though in particular in the experiment it cannot be excluded that some components are transferred also by the walls. Since the process has a logarithmically slow dynamics, the rule-based models [62, 63, 64] have been preferred in computational investigations in order to cover large time scales like cellular automata have been used. They nevertheless have the problem that physical parameters are difficult to incorporate, while basic physical principles, like Newton’s “action equal to reaction” principle cannot be applied here. Only recently, discrete element simulations also have been used [65]. Due to the slowness of the dynamics, analytical investigations have been undertaken [66] despite the complexity of the system. One point which should not be forgotten about granular compaction with tapping is that the long-time limit is not necessarily the densest packing available for a given kind of material: For some materials, higher packings have been obtained by first evacuating the vessel with the granular filling and then letting the air stream in [67]. This also shows that the phenomenology of fluid-immersed particle systems may be considerably more rich than for dry system, even if only the effect on the particles is concerned.

6.1.2 Outline of this Study

We are interested in the difference between dry and fluid-immersed particle systems in the short time dynamics when a relatively viscous fluid ($\mu_f = 1[\text{Pa} \cdot \text{s}]$, thousand times the viscosity of water) is introduced into the pore space. While conventionally round granular materials are used in the context of tapping, for our polygonal particles reordering via rotation and rolling is hardly possible, so a different phenomenology can be expected. We limit ourselves to the initial time of the compaction as the numerical solution of the flow field is rather more costly than the DEM-simulation. Nevertheless, as there are hardly any codes for microscopic simulations, we can obtain informations about the system which are hardly accessible otherwise. Moreover, we can try to extrapolate to long-term limits by reducing or altogether switching off the static friction, as will be shown in Section 6.1.5.

Introduction of viscous fluid into the tapping experiment can lead to two possibilities:

1. One can imagine that it will increase the damping in the relaxation process. Under these conditions, the dynamics would be slowed down even further.
2. When particle do not push only neighboring particles, but also buoyancy and inertia of the surrounding fluid transmit the tapping impulse, an improvement in the transmission of the excitation through the granular column can be expected.

The question is which element will gain the upper hand: The additional damping and lubrication by the fluid or the enhanced transmission. For this problem, our simulations allows to set the idealized parameters, while an experiment would be affects by changes in the surface chemistry and friction.

6.1.3 Initialization and preparation of the System

The particle shape is constructed using the first method discussed in Section 2.2.1: By inscribing regular polygons into ellipses (semi-major axis of 1.2, semi-minor axis of 1, diameter of 5 [mm]) and randomizing the shapes and sizes by changing the radius rays for the corners randomly by $\pm 10\%$. Only convex particles are used. The corner numbers of the particles range between 5 and 9. The initial configuration of the granular packing is constructed by dropping the 195 particles in the dry DEM-simulation from initial positions with the center on a regular grid (Fig. 6.1 (a)) and

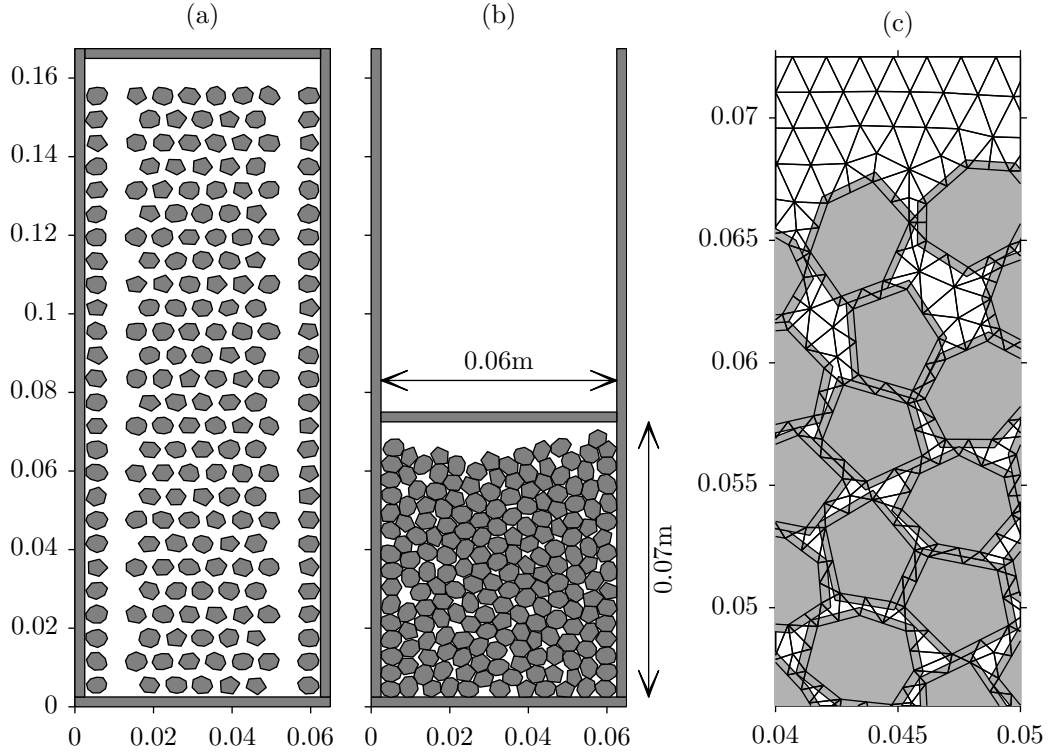


Fig. 6.1 (a) Initialization of the system with 195 particles. (b) system with the particles settled down which is used as initial configuration. (c) Enlarged region of (b) showing the pore space and triangular meshes around the particles.

wait until the particles have settled down and the vibrations in the agglomerate are damped out (Fig. 6.1 (b)). In the initial grid (Fig. 6.1 (c)), a stripe of particles is removed near the left and right boundary to allow the development of stronger disorder than what is possible if the positions are occupied regularly. Then the resulting packing is used as the initial configuration for both the dry and the immersed system.

The friction coefficient of $\mu = 0.3$ (both for the static and dynamic friction, both between particles as well as between particles and walls) is used for the dry system as well as for the system of immersed particles, to clear up the differences in the dynamics which come from the introduction of the fluid. For many particle materials, the friction coefficient of the fluid-immersed particles would probably be lower. The Young's modulus is $Y = 10^6[\text{N/m}]$ (respective to a depth of $1[\text{m}]$), the damping constant is 1.5. The Young's modulus may look small compared to the ones for three dimensional materials like stone, which are of the order of several hundred gigapascal but the sound propagation occurs by the contact of particles which in the physical reality are microscopically rough, i.e. only the contacting surface asperities lead to the propagation of sound: In that respect, our Young's modulus is meant to model

an averaged, smooth surface which is softer than the corresponding asperity-filled surface of actual experimental materials. The density of the granular particles is $5000[\text{kg}/\text{m}^3]$ (again, respective to a depth of $1[\text{m}]$), while the bulk density (including the pore space) varies between $4195[\text{kg}/\text{m}^3]$ and $4280[\text{kg}/\text{m}^3]$, e.g. the porosity of around 0.39 (including particle shadows), 0.16 (without the particle shadows). It is not possible in the current code to fill in fluid as in physical systems gradually, so we have to submerge the system into the fluid all in one. Accordingly, this would lead to a fast change of interparticle forces due to the effect of the buoyancy. To offset this effect, for a fluid with density $\rho = 1000[\text{kg}/\text{m}^3]$ and a final target density of the grains $\sigma = 6000[\text{kg}/\text{m}^3]$, we set up the particles of the dry system with a density $\sigma - \rho$: When the system is reinitialized with fluid, the equilibrium positions for the dry systems are then exactly the equilibrium positions of the particle system with fluid.

6.1.4 Tapping

Tapping (i.e. inserting a shock/pressure wave from the bottom of the system) has the dual effect that there is a sound wave propagated through the system while the particle positions are perturbed in such a way that a positional reordering becomes possible. For modeling the tapping, i.e. conferring an impulse to the particle system via the boundaries, we have several possibilities. The tapping can be implemented as

1. a displacement of the boundaries,
2. or a specification of a velocity of the boundary (without displacing the boundary at all),
3. or as a combination of both.

Because sudden displacements of a neighboring particle can lead to very large shocks, we opted for keeping the wall position constant and change only the (dummy) velocity on the walls surface. Next we have to decide which boundaries we want to manipulate: We can tap either only a part of the boundaries, i.e. the floor, or all the boundaries. In physical systems, tapping the floor of a cylindrical vessel will lead to a propagation of the shock also along the cylinders walls: To transmit momentum only through tapping of the bottom, it would have to be unconnected to the cylinder's walls. Because very often it is not clear in the experimental systems how the tapping affects the bottom system, we decided to investigate both possibilities:

Tapping of the bottom was used so that the upward propagation of the shock wave through the system could be measured (Section 6.1.6), and the differences in shock propagation with and without fluid could be identified. Tapping of the whole boundary (Section 6.1.6) was used to enforce a more physical macroscopic settling of the granular matrix.

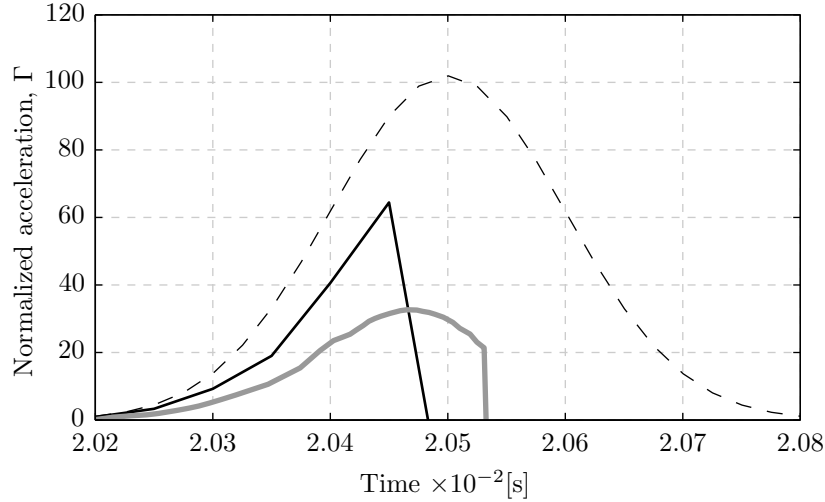


Fig. 6.2 Time evolution of the acceleration of a particle in the lowest layer of the dry simulation (black) and wet simulation (gray) as well as intensity of the original pulse (dashed line).

The pulse is constructed from a Gaussian

$$f(t) = a \exp \left(-\frac{(t-b)^2}{2c^2} \right), \quad (6.1)$$

where the amplitude of the pulse a is chosen as 100, $b = 5 \times 10^{-4}$ and $c = 1 \times 10^{-4}$, see Fig. 6.2. The time between the beginning of one pulse and the beginning of the next is 0.4[s]. Together with the magnitude of the pulse, we rather quantify the response, i.e. we give time evolution of the acceleration for a particle in the lowest layer. The ratio between average acceleration by the pulse and gravitational acceleration ($9.81[\text{m/s}^2]$) was $\bar{\Gamma} = 17.3$, with a maximal value of $\Gamma_{\max} = 64.4$. While for the system in fluid, the ratio was $\bar{\Gamma} = 18.8$, with a maximal value of $\Gamma_{\max} = 32.7$. As can be seen in Fig. 6.2, the acceleration of a particles contacting the wall is not exactly equal to the acceleration of the wall itself: The response for the dry system is only a part of the upward slope, while the the system with fluid, the particle accelerations is proportional beyond the maximum of the Gaussian, but in both cases, the amplitude is smaller.

6.1.5 Evolution of the center of mass

We monitor the evolution of the center of mass over time as the parameter of the relaxation, rather than the density: As our system does not have many particles, the uppermost layer consists of relatively many particles, compared to the total system. Therefore, it is difficult to define the upper boundary of the system and therefore the density with respect to the upper layer. To avoid ambiguities, as different particles at the upper layer may be displaced at different speeds (some might even be displaced upward), and to be able to compare systems of different dimensions, we use the position of the center of mass scaled to 1 as the parameter which is observed for the settling.

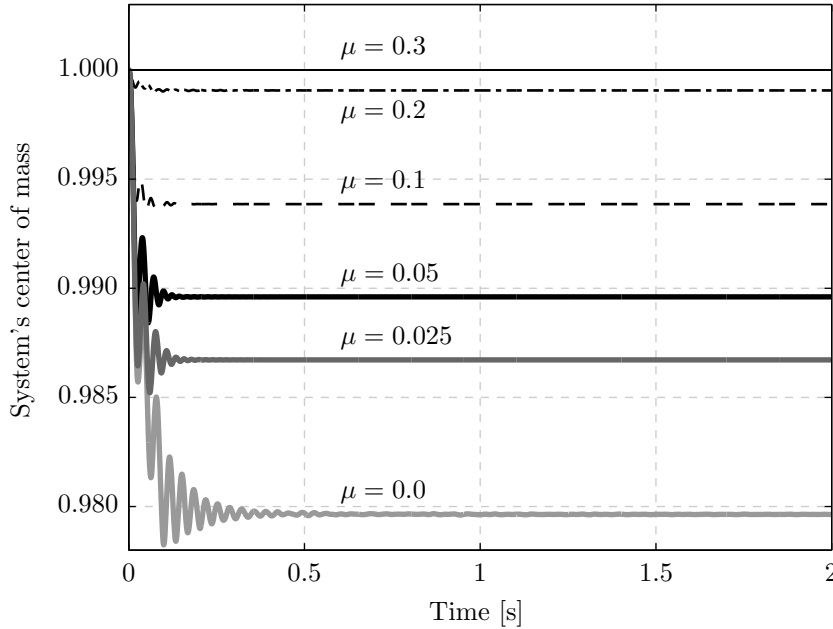


Fig. 6.3 Position of the center of mass for reducing the coefficient of friction μ .

Vibration of particles in sliding contact is known to lead to continuous slipping (“creep” [18]). Therefore, we computed the relaxation of the dry system without fluid from the initial configuration when the friction coefficient is reduced from 0.3 to lower values down to 0.0, see Fig. 6.3. The packing density increases (i.e. the center of mass is lowered) monotonously with lowered coefficient of friction. This should give a plausible long-term limit, as vibration or tapping leads to a momentary reduction of the contact, which allows slipping, as would a reduction of the friction coefficient. The packing for vanishing static friction with coefficient $\mu = 0.0$ is therefore a limiting case for the packing density, though it may not be actually reached, depending on the parameters (amplitude, frequency) of the tapping pulse. Fig. 6.4 shows that the the center of mass is lowered not according to a logarithmic

dependence, but according to a power law. Additionally, one can see in Fig. 6.3 that the vibration is damped stronger if the friction coefficient is high. Why one could say that this should be predictability so, one should remember that our friction law is not exact, but a model (see section 2.1.3): That the damping improves with the friction model shows that the model is physical. With respect to the two energy dissipation mechanisms in the contact model one can say the following: The effect of the normal damping in Eq. (2.7) is weaker than that of the Coulomb friction from Eq. (2.16).

6.1.6 Results

Tapping of the Bottom Only

The shock wave induced by the tapping on the bottom is shown in Fig. 6.5 as the height of that particle which experiences the strongest dislocation from one time-step to the next versus time. In principle, this way of plotting actually shows the propagation of the sound wave, and the maxima of the sound wave at different times allows to identify the sound velocity. Nevertheless, there are several particles at different height are only weakly connected themselves in the granular matrix, so that their rattling inside a “cage” of particles which are stiffly connected in the matrix is hardly damped, especially for the system without fluid (strings of the same symbols at different height in Fig. 6.5). The sound wave in the dry system propagates with a speed of 2.36 [m/s], while in the immersed system, it propagates with 3.68 [m/s]. The sound velocity of the continuum material would be 14.1 [m/s] for the dry and

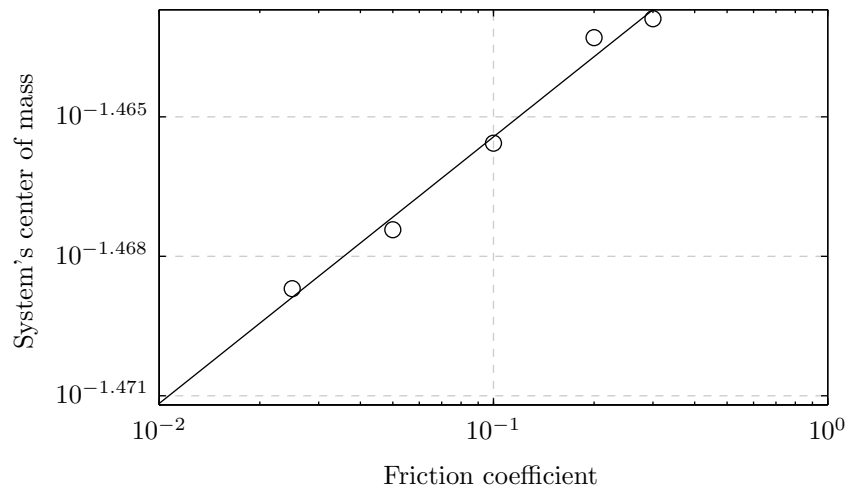


Fig. 6.4 “Final” potential energy of the wide system with different coefficients of friction μ in double logarithmic plots (○) and the fitting (solid line).

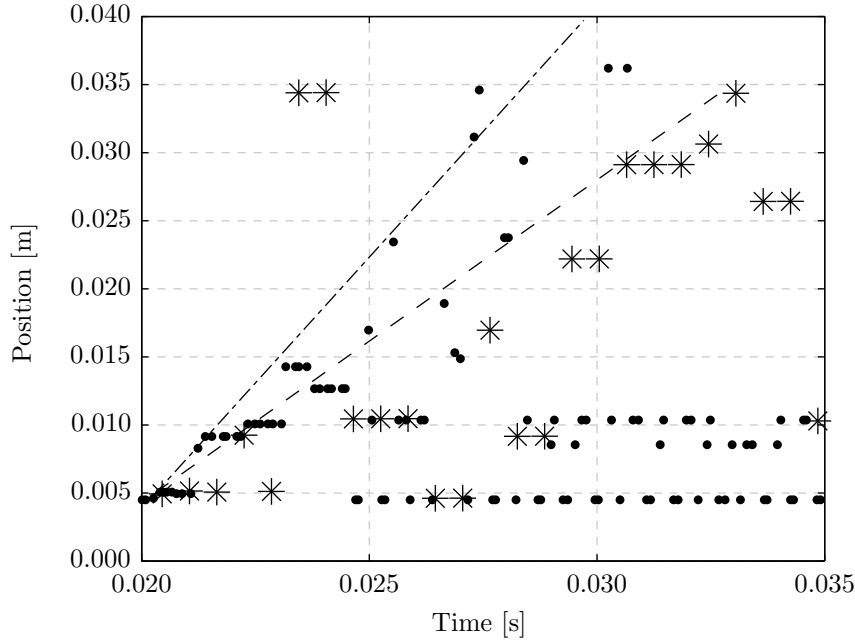


Fig. 6.5 Shock propagation through the system for the dry (*) and the immersed system (●) for tapping of the bottom. The symbols denote the position of the particle with the maximal dislocation over the time of the vibration pulse (see Fig. 6.2) for a given time-step. The wavefronts for the immersed system are indicated by the dash-dotted line, for the dry system by the dashed line to guide the eye.

12.9 [m/s] for the immersed material due to the density difference necessary due to the initialization. We repeat that the fluid is incompressible, while the pressure wave propagates only through the granular matrix. The corresponding fronts of the sound waves are indicated in Fig. 6.5 with dash-dotted (immersed system) and dashed (dry system) lines. The fluid immersed system shows a significantly higher sound velocity than the dry system. This means that the surrounding fluid helps to stiffen the interparticle contacts (see Fig. 6.6) and speed up the propagation speed of the shock wave. The sound wave is so much higher than in [5], because due to the compaction on the bottom, the particle contacts are pre-stressed and therefore much stiffer than on the surface.

The compaction results for the system which is tapped on the bottom are

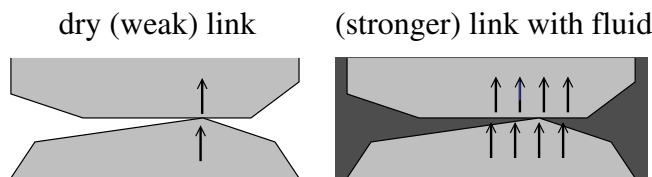


Fig. 6.6 The incompressible fluid helps to stiffen the interparticle contacts and leads to higher sound velocity.

shown in Fig. 6.7. The vibrations are damped more strongly (actually, practically overdamped) in the fluid than in the dry system. While initially, the centers of mass of both the dry and the immersed particle system started out at 1, the center of mass of the dry system falls faster, as can be seen by the increasing distance between the gray and the black curve in Fig. 6.7.

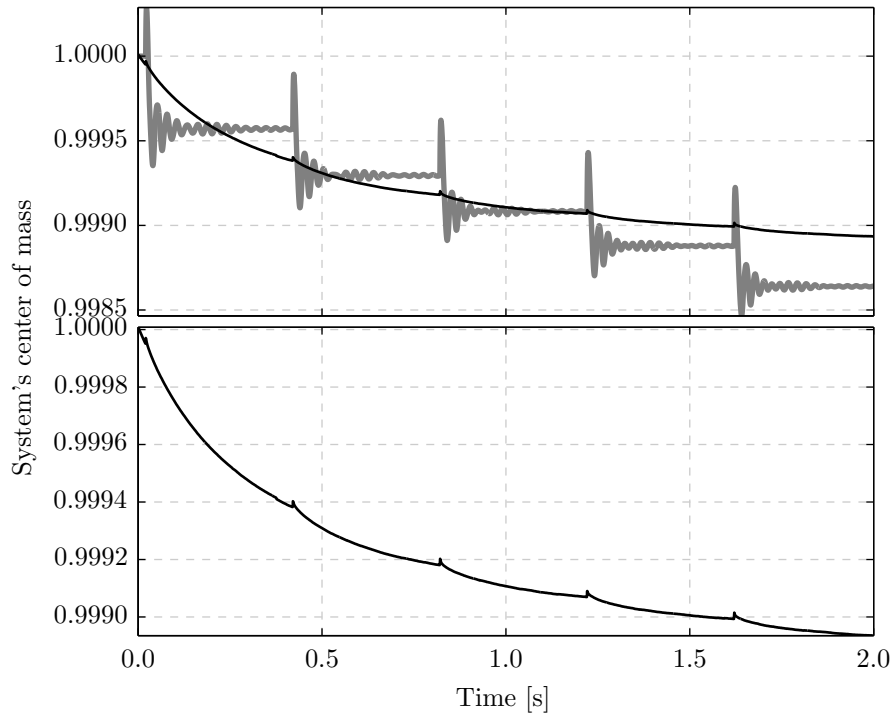


Fig. 6.7 Compaction of system with tapping only the bottom for the dry (above, solid line in gray) and the immersed particles (above, solid line in black), as well as the latter zoomed (below).

Tapping of the Whole Wall

The compaction results for the system for which the tapping is felt at the whole wall are shown in Fig. 6.8. The initial response amplitude for the system with tapping of the whole boundary is about 11 times as large than for tapping with the same intensity on the bottom only. The compaction is consistent with the previous section. As in the case of tapping of the bottom, when the whole boundary is tapped, the center of mass in the dry system falls faster: With both systems' initially centers of mass of at 1, the gray curve for the dry system separates from the black curve for the immersed system in Fig. 6.8.

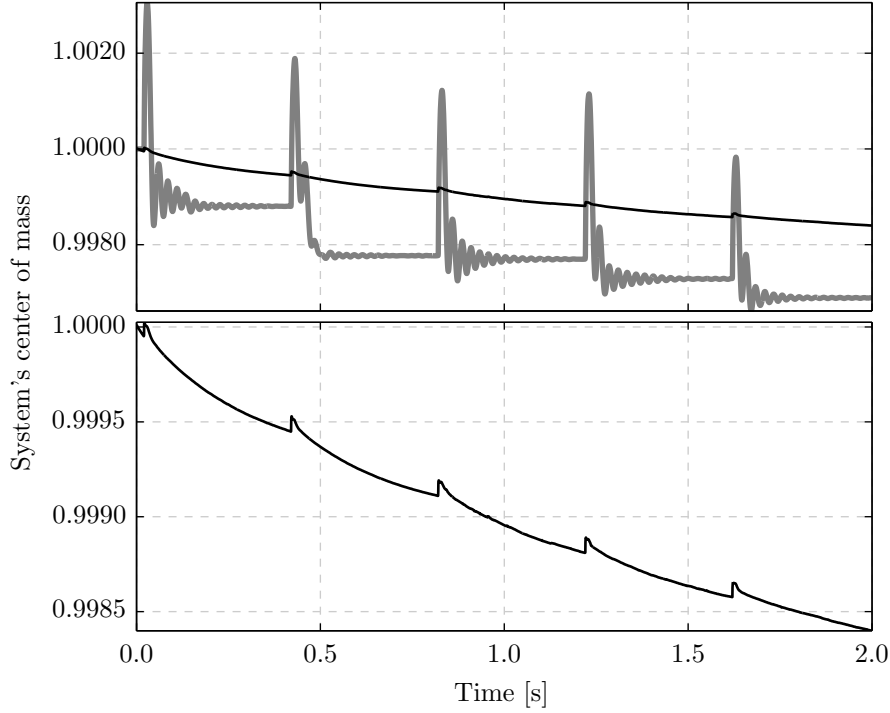


Fig. 6.8 Compaction of system for with tapping the whole wall for the dry (above, solid line in gray) and the immersed particles (above, solid line in black), as well as the latter zoomed (below).

6.1.7 Summary

Our simulation has shown that the addition of (an incompressible) fluid to a granular assembly can increase the sound velocity in the system, compared to the dry case. The introduction of fluid into the slow dynamics of compaction of granular particles via tapping even for our single parameter with only one value for the density and viscosity showed a considerable variation of effects. For the dynamics of the settling of the center of mass, introducing the fluid makes the settling slower, even though the sound propagation is faster.

6.2 Collapse of Granular Column in Fluid

The following research was the article [68] where collapse of a granular column in a viscous liquid is experimentally investigated. We tried to simulated a similar system, but mostly due to problems with the high computational effort in the fluid domain, we could not come close to the particle number of the experimental system, which for the two-dimensional case would have been of the order of 16000 particles. We simulate the collapse of granular column in fluid and compare it with the system of

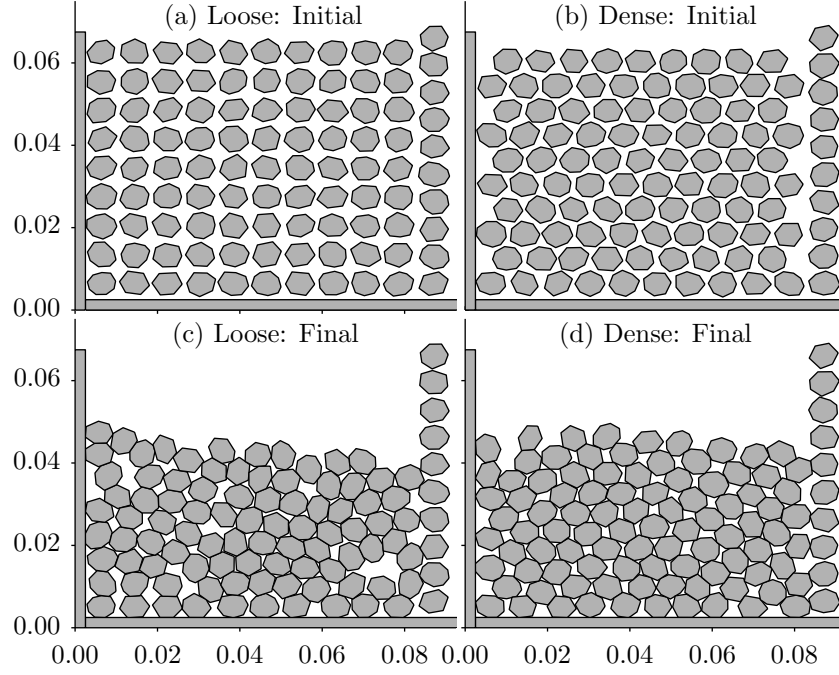


Fig. 6.9 Preparation of the granular columns with loose packing $\phi = 0.767$ in (a) and (c), as well as dense packing $\phi = 0.793$ in (b) and (d). After the preparation, the step with the particles is released by removing the non-contacting particles on the right.

the same configuration without the fluid. The convex particles are also constructed in the similar way as in the previous section by inscribing regular polygons into ellipses (semi-major axis of 1.2, semi-minor axis of 1) and randomize the shapes and sizes by adding random numbers of $\pm 10\%$ of the radius to the corners. The numbers of corners of the particles range between 6 and 9. As the experimental paper [68] used two different granular structures, a loose and a dense one, we decided to use these different packing densities also in the simulations. To obtain columns with different bulk densities as in Fig. 6.10, the granular columns are constructed by dropping the particles in the dry DEM-simulation in different initial configurations:

- (a) For the loose column, the particles' initial configuration are arranged so that the centroids are positioned on a square grid as in Fig. 6.9 (a).
- (b) On the same square grid arrangement, particles' with 80–120% of the original radii are used to produce system with larger size dispersion.
- (c) For the dense column, the centroids are positioned on a hexagonal grid as in Fig. 6.9 (b).
- (d) Then, on the same hexagonal grid arrangement, particles' with 80–120% of the original radii are used to increase its size dispersion.

The resulting volume fractions ϕ are shown in Fig. 6.10 (a)–(d) respectively.

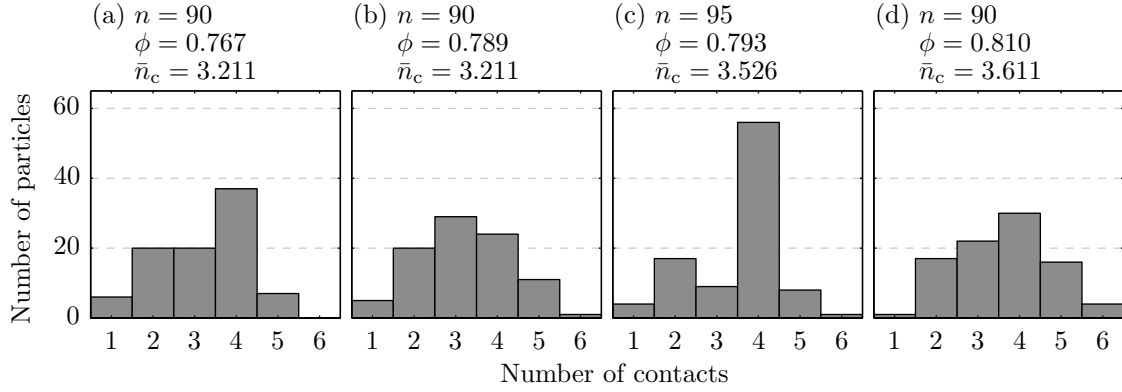


Fig. 6.10 The distribution of the number of contacts \bar{n}_c for the granular columns with different volume fractions ϕ and number of particles n .

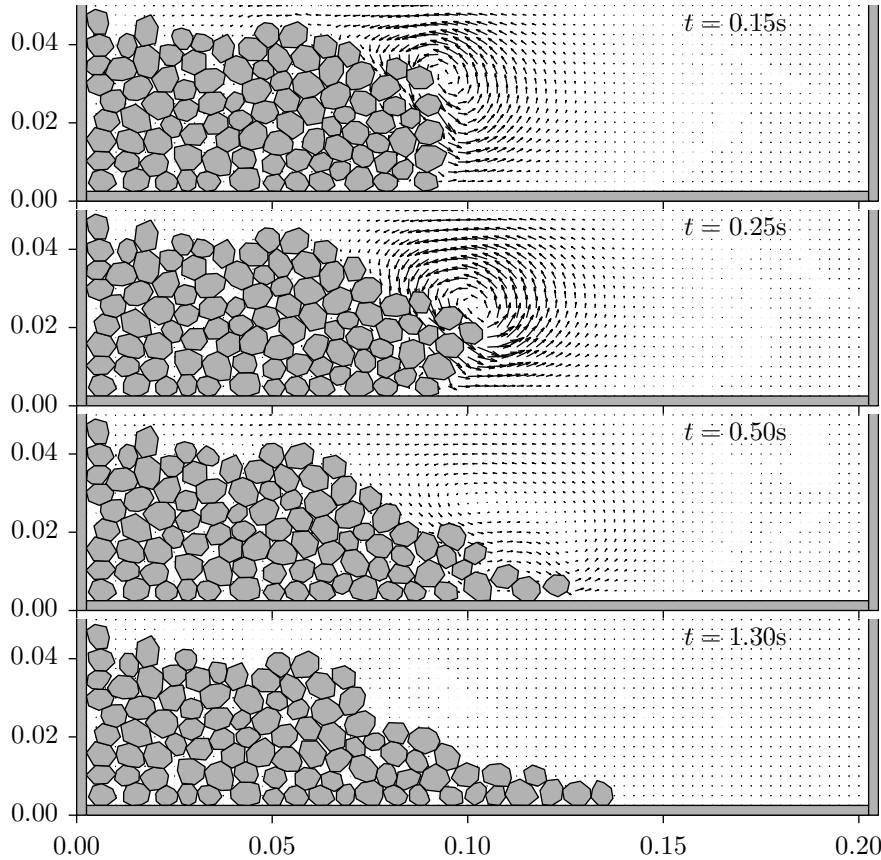


Fig. 6.11 Snapshots of the collapse of granular column in fluid for system with volume fraction of 0.789.

A column of particles on the right side is fixed as a wall, see Fig. 6.9. The remaining particles which are supposed to move are released from the initial positions and wait until all the particles have settled down. Though the initial positions from which the particles are dropped differs from square to hexagonal, the resulting

packing does not reflect this initial configuration: The volume fractions differ only by 5.3% and the contact numbers by 10.3% (see Fig. 6.10).

After the particles have settled down, the non-contacting particles which were previously fixed on the right side of the column in Fig. 6.9 are removed to release the granular column at $t = 0.1$ [s]. Both the dry and the immersed system are conducted from the same initial particle configuration (position, orientation). As mentioned in the tapping section, the dry system is simulated with particle's density of 5000 [kg/m³], while the immersed system with particle's density of 6000 [kg/m³], inside a fluid with density 1000 [kg/m³] to offset the effect of buoyancy. The snapshots of the collapse of granular column for the system immersed in fluid with volume fraction of 0.789 is shown in Fig. 6.11.

The maximum position of the front over time is given in Fig. 6.12. The decrease in the final positions of the fronts in the immersed systems is due to the damping effect from the fluid on the rolling motion of the particles. This shows that for our system immersed in fluid, rolling of particles becomes less important than for the corresponding dry systems. Nevertheless, the small number of particles for such system makes comparisons with experimental results difficult. Rondon et al. found experimentally that a granular column made of glassbeads will show a faster collapse for loose packings, and a slower collapse with dense packings [68]. Beyond

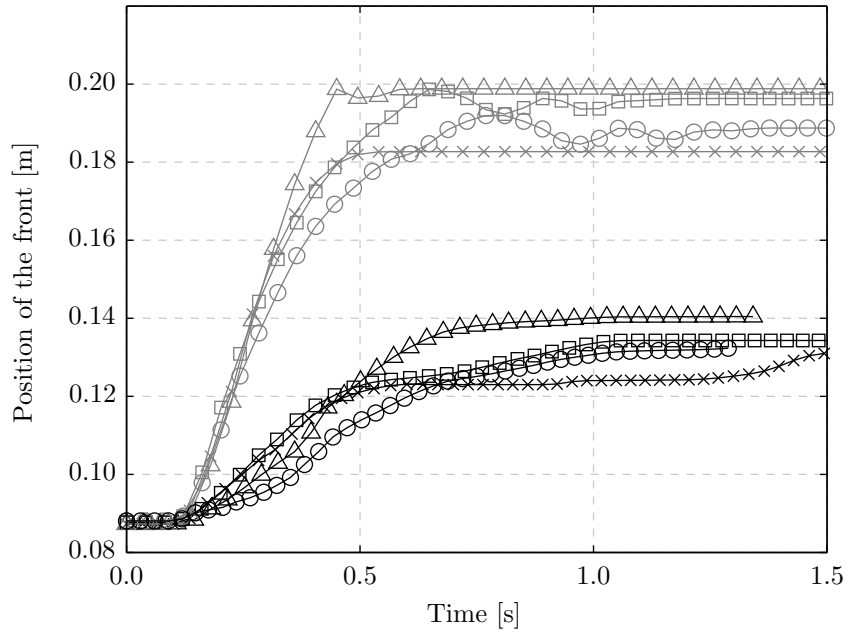


Fig. 6.12 Time evolution of the front position for the dry system (gray) and the system with particles immersed in fluid (black) with different volume fraction (\triangle : 0.767 , \blacksquare : 0.789 , \times : 0.793 and \circ : 0.810). The oscillatory motion of the wave front for the dry case comes from the rolling forward and backward of the foremost particle.

the article [68], there is also a video where one can see that Rondon's mobility of the granular front is partially due to secondary avalanches which form on steps which resulted from the initial decay of the front. In our numerical experiments with smaller number of particles and only small size dispersion the advancing fronts are rather independent to the volume fraction. In the current state of the research, it cannot be decided whether the reason that the result is mostly independent of the initial density is the small number of particles, or the dimensionality (two dimensions instead of three).

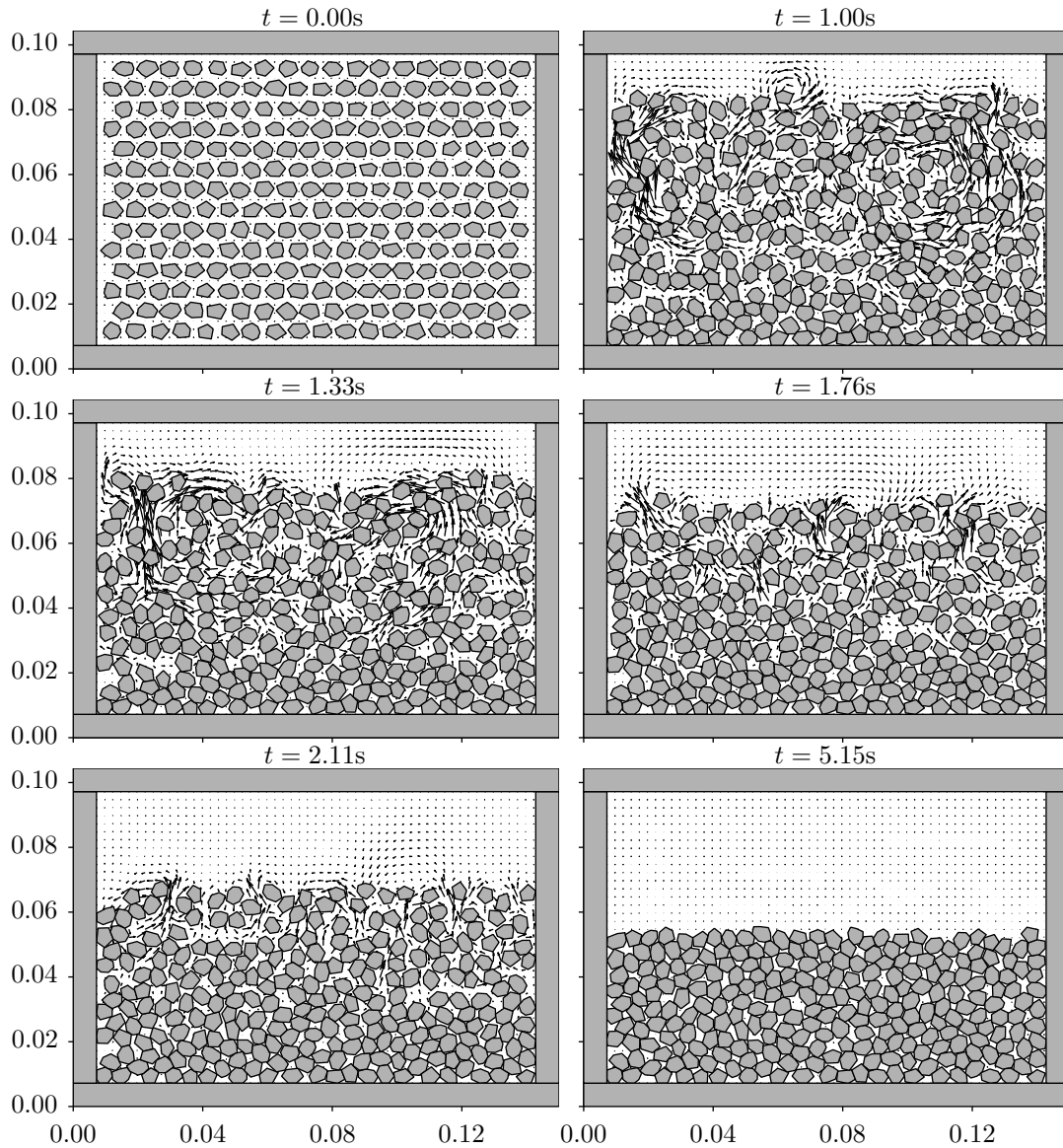


Fig. 6.13 Sedimentation of 252 polygonal particles of different shapes in a fluid domain of size 135.6×90 [mm] with about 3700 triangular elements.

6.3 Sedimentation of Multiple Particles in a Fluid

Sedimentation is the common term for the the deposition sand settling of granular materials in fluids. In Fig. 6.13, we shows the snapshots of the simulation of 252 particles (radius of about 3.0 [mm]) sedimenting in the fluid ($\mu_f = 1 [\text{Pa} \cdot \text{s}]$, $\rho = 1000 [\text{kg}/\text{m}^3]$) using the same approach as the above. The Reynolds number of the system is up to $Re \approx 4.2$. This shows that our code is able to handle more particles, with arbitrary convex shape. A single run of the immersed system took up to four weeks in a dual-core Mac mini. The most expensive part was the computation of the free fluid, which was resolved by triangles about 1/12th of a particle area.

Chapter 7

Epilogue: Beyond the Scope of this Thesis

In this chapter, limitations and possible improvements of the current program are addressed. Necessary extensions, which affect performance and stability, as well as desirable and possible extensions, and further possible applications of the code in the current form and with smaller and large modifications are discussed here.

7.1 Performance improvements for future simulations

Grid coarsening The main limitation in the current form is that it is not possible to coarsen the grid of the fluid domain automatically. This means that even in regions where the flow field varies only marginally, so that much larger grids could be used, a grid size similar to the one with strongly varying flow is necessary. While automatic re-meshing towards smaller grids for better stability and accuracy is implemented and works, the mesh cannot be coarsened beyond a standard size of meshes. This is related to the fact that the mesh size at the boundary cannot be set automatically, which basically limits the triangle size. This is a big obstacle performance-wise, as the computation for “dummy regions” of the flow which serve only to put the boundaries far away consume considerable CPU-time.

Parallelization of the solver Currently, the MATLAB-code uses the UMFPACK-package [35] for its built-in sparse solver, which is very efficient even for badly ordered systems. While efficiency for a single core is quite high, the

bottleneck comes with the parallelization: While setting a larger number of cores is possible, it only increases the CPU-usage for the system, while the speed of the algorithm does not increase. This means, the algorithm looks for possibilities for faster solution and distributed execution, but does not find any. Attempts to use Krylov-space solvers, whose iterative matrix-vector multiplications allows straightforward parallelization, turned out impracticable: The convergence was so bad, even for GMRES and the newest, most powerful member of the family, BiCGSTAB(l), that nowhere the necessary accuracy for the continuation of the Newton-iteration could be reached. Even using the first digits of the numerical solution as starting value did not improve the convergence. We have to conclude that the equations are scaled so badly that the usage of Krylov-solvers is not possible. Using straightforward preconditioners did not improve the convergence. Beyond UMPFPAK, very few alternative codes for sparse solvers exist. Other parallelized multi-frontal codes are the MUMPS-package [69] and the PARDISO-package [70] and it would be desirable to implement them (there are also MATLAB-interfaces) and benchmark them against UMPFPAK. In the end, this means that improvements in the parallelization of the algorithm must be left to the developers of the sparse linear algebra packages.

Parallelization via domain decomposition Domain decomposition is a standard method for many explicit solvers which use relaxation methods. Unfortunately, for the finite element discretization and the non-linear systems which occur in our simulations, no mathematically exact domain decomposition (local computation of the solution on independent subdomains and modification of the boundaries using additional computations and global communication). What seems feasible is nevertheless the geometrical distribution of domains with stationary flow on the boundaries onto different cores. This would at least speed up the computation for initial phases of the problem, and for problems with non-violent flow changes.

7.2 Problems which can be tackled by the program in its current form

Apart from performance- and system-size issues, the following problems can now be tackled by the simulation:

Damping of foundations under water A current problem in the field of civil engineering with respect to foundations is that there is no method to a priori compute the effect of groundwater or saturated water on the damping of foundations. In general, computations are done via finite element methods where the damping parameter is introduced ad hoc. Accordingly, settling of foundations under vibration is a common problem which affects building (e.g. pillars for fast train tracks) which have been unaccounted for in the planning stage [71]. With our current problem, we can investigate settling effects and damping effects based on assumptions on the grain interaction.

Brazil-nut effect for liquid-fluid mixtures The rising of larger particles among smaller particles (“brazil-nut effect”, “size segregation”) in dry granular materials happens gradually under continuous excitations: Reordering due to shaking or convection causes the smaller particles to move below larger particles, so that the larger particles rise. Buoyancy plays no role in this case, as even particles with higher density (steel) may rise above particles with lower density (glass beads).

In fluid-saturated media, a much more violent rising of lighter larger bodies over heavier granulates can be observed: The head of plastic pins rise abruptly over the surface of layer of loosely deposited sand they were buried in, due to mere tapping, without continuous excitation. In this case, the pin heads have indeed lower density than the surrounding sand. The interplay between granular reordering and fluid dynamical effects, including buoyancy is not yet understood. This phenomenon may be relevant for the fluidization where during earthquakes, sewage ducts start rising in the streets, blocking traffic and rescue effects: In both cases, the triggering mechanism is of relatively short duration, while the reordering which follows (including fluidization) will continue when the excitation is already over. The interplay between the granular matrix, the buoyancy (which may actually be the energy source for the processes) and the flow in the pore space can be investigated at least in principle with our program.

Fluidization of fluid-saturated granular materials As for the rising of particles, fluidization due to external vibrations can be investigated with the code in the current form. Fluidization may be a bit more difficult, as the reordering of the granular matrix will be over a wider area and the densities will vary more strongly: this poses higher demands on the grid generation, which may need an extension beyond the current stage of the algorithm.

Flow problems in slopes Calculated over decades as average, land slides due to strong rains cost more lives in Japan than earthquakes. While in principle the explanation is that "the landslide was triggered by strong rain in the area, the slopes near the landslide can be supposed to have suffered exactly the same precipitation, while no landslide occurred. In other words, the real reason for landslides, which can be suspected to be an interaction between fluid and soil, is still unclear. One can imagine that changes of pressures on the soil due to the filling of the pore space with water may lead to shear bands which provide subterranean channels for the water which hollow out the soil even further. The sudden appearance of springs during strong rains on slopes is considered a sure sign of danger: This may be flow in shear bands, nevertheless, the mechanism for that is not understood either. Such scenarios can be microscopically investigated with out code at least in two dimensions.

Biaxial compression without fluid While biaxial compression with the granular part of the simulation has been performed years ago [72] for the case of dry particles. Nevertheless, for many technical problems not this "drained" experiment, but the undrained case is relevant where the pore space is filled with water. Up to now there is no micromechanic understanding how lubrication effects and modified inertia will modify the strength behavior.

7.3 Desirable extensions for future simulations

For some applications, some limitations of the simulation code have already been noticeable. In the following, some desirable modifications for the programs are listed which can be implemented with predictive programming effort.

7.3.1 Extensions which would improve speed, accuracy or stability

While the simulation code was written over a period of six years, it was nevertheless not possible to include various desirable features which could help to improve speed and stability due to relatively small modifications of the program in its current form. Nevertheless, these extensions need a more profound understanding of the related mathematical issues.

Field-dependent grid-refinement While the algorithm is already beyond the test-stage, a field-dependent grid refinement can be implemented. This is done

by “stiffening” the springs which consist the edges of the triangles, according to the absolute value of the fields, respectively their gradients. This means that vortices moving inside a domain of stratified flow can be simulated with smaller gridsize and therefore, better accuracy.

Periodic boundary conditions Periodic boundaries are rather difficult to realize for finite element methods. For finite difference methods, the field amplitudes are only defined at the stencils, so that with suitable offsets, arbitrary jumps in the field can be introduced. Accordingly, it is very easy to create a pressure gradient so that periodic pipe flow flow is introduced. For finite elements, the fields in the whole domain are given via piecewise linear polynomials, so that the flow is continuous. The constructs which are necessary to obtain periodic boundaries with a linear increase of the pressure are explained in Gresho et al. [28, pp. 707–712].

For spatially periodic flows in x -direction, it would be easier to set the flow velocities on the lower (and/or) upper boundary and set the connectivity of the nodes for the elements so that the grid points on leftmost boundary are simultaneously the grid points on the rightmost boundary. In that case, no step-like change of the pressure would be necessary.

Improvement of the convergence criteria Currently, the Newton–Raphson iteration uses the absolute change of the solution Eq. (3.72) as convergence criterion. While both pressures and velocities occur in the equations, and because the units are in SI-units¹, the numerical values for the pressures and the velocities have altogether different magnitudes. Currently, relative weighting of pressures versus velocities is not implemented. A more subtle treatment of the convergence issue may improve the speed of the simulation at least in those regimes where the flow does not vary too much.

Stabilization by projection The numerical integration step Eq. (3.63) is a self-consistent solution of the finite element equations for the velocities under the constraint of incompressibility, which leads to pressures as Lagrange parameters. While the equations are solved via Newton–Raphson iteration, so that the solution fulfills the incompressibility according to convergence criteria, it is still possible that the solutions violate the incompressibility condition.

¹The use of dimensionless units makes no sense, as the fluid system has an all together different set of dimensionless units than the granular system, so the need to exchange the forces between both systems and to verify the physicality with experiments makes SI-units the most attractive choice.

An additional evaluation of the error, i.e. the degree to which the solution violates the incompressibility may be a useful tool to determine convergence, respectively to adapt the time-step.

A violation of the incompressibility seems to be in particular a problem for fast moving particles in narrow channels for the fluid. Problems with the violation of constraints for differential algebraic equations are nowadays usually dealt with via stabilization by projection [73], which means that only that part of the solution is used which is in the “constraint manifold”, i.e. the mathematical space which fulfills the constraint. Such an additional projection step may be necessary to improve the stability of the simulation in particular for violent particle motion, i.e. for simulations of land slides and the evaluation of damping effects of the fluid in soils on foundations under sudden external load.

7.3.2 Extensions which make additional phenomenology accessible for simulation

Hydrophilic or hydrophobic character of particles It turns out that e.g. sinking velocities of particles depend on the hydrophilic or hydrophobic character of particles relative to the respective character of the surrounding fluids. Particles with hydrophobic coating sink fastest in water. [74]. While in our current simulation, the hydrophilic or hydrophobic affinity is not implemented, it would be easy to include forces acting on the boundary additional to the pressures and drag on the surfaces.

Free surfaces with surface tension For the free boundaries in Chapter 4, the evolution is currently only according the velocities on the surface. Surface tension can be introduced by specifying forces (pressures) on the nodes of the surface which are computed based on the curvature of the surface. Due to the correspondence between third order splines and the elastic tension of rods due to curvature. Accordingly, surface tension can be computed as a pressure proportional to the curvature term of splines which connect neighboring surface elements. A test version of the program showed indeed a faster vanishing of wrinkles on the surface of the fluid.

Wetting and non-wetting character of surfaces While surface tension can be understood as the curvature described by the spline connecting neighboring surface nodes, the wetting property can be understood as the slope of the splines on the node which lies on the solid surface which consists the fluids

boundary: According to the character, the contact angle for that slope can be between 0° and 90° (high wetting) or between 90° and 180° (low wetting). While we have tested an implementation of splines which are computed by the support points, other mathematical forms exist which take the slope at the end into account [75]. The physicality can be tested by the rising height and the shape of fluid menisci in communicating vessels.

Contacting multiple fluid surfaces Contacting multiple fluid surfaces, such the collapse of a breaking wave so that its tip reaches again the original fluid surface, is computationally rather problematic: For the current state of the program, there is no remeshing included for the case that a simply connected becomes non-simply connected. Further, a collision detection is necessary to determine where the contact between the approaching fluid fronts closes. The advancing of the fluid front itself should be treatable with the methods described in Chapter 4. Also, computation of the fluid flow itself for non-simply connected FEM-domains itself will not pose any problems, as that is the geometry for which the program is written anyway. When the issue of the grid change is solved, from the point of fluid equations, an empty cavity can collapse without any resistance.

Free surfaces for particulate surfaces While free surfaces of fluids are well mathematically well defined, for assigning a fluid surface to a rough granular surface one is on relatively less save ground. While in principle it should be possible to work with adhesion and surface tension so that the fluid surface on a particle bed forms by itself, it is not clear whether the resulting surface might not contain inclusions which do neither belong to the granular nor to the fluid space, but are filled with “air”. Such multiple contact problems for are computationally and algorithmically certainly very complex. It would be easier to define a fluid surface on phenomenological grounds, nevertheless, if there are crevices between surface particles, it is difficult to give necessary conditions for the surface. Probably the easiest way to define the surface will be to define it with a fit to the granular slope along a line through the centers of mass of the surface particles: For that approach, there is already experience with the necessary algorithms [76].

7.4 Possible extensions

In the following, some possible extensions are listed which are obvious due to the program structure. While the necessary programming effort is rather speculative, it is nevertheless the minimum of what can be expected in terms of necessary work.

Gas-Fluid simulation, Bubbling In principle it should be possible to simulate bubbles inside the fluid as empty spaces where the surrounding fluid boundary has some surface tension, while the pressure of the gas domain is added as pressure on the fluid boundary.² As long as the density of the gas is much smaller than that of the surrounding fluid, the flow processes in the gas can be neglected.

Fluid-fluid simulation For two different fluid domains, the resulting equations in Eq. (3.63) will become block-diagonal. The interaction between both fluids will then be via the pressures on the boundary between the fluids. Simulations with two fluids will be most interesting if one fluid is hydrophilic and the other hydrophobic. Accordingly, there will be surface tension and repulsive forces between the different fluids acting on the boundary between the domains. As the velocities calculated with the FEM-method are a result of both inertial, internal and external forces, it should again be possible to advance the boundaries with the velocities which are computed on the boundary via the FEM-code.

Gas-fluid-fluid simulation The exploration of oil- and gas-fields deals with the problem that there are usually hydrophilic (water) and hydrophobic (oil) layers, together with layers of hydrophilic and hydrophobic rock, as well as domains filled with gas. The interaction on the boundaries would be determined between the respective forces between the fluids, respectively the forces between solids and rocks. The actual practical problem for such simulations is the large number of parameters which have to be included in the simulation: 2 for the fluid-solid boundaries, and 3×2 for the fluid-fluid boundaries, including 3 parameters on triple boundaries.

While the particle number would be much too small if for the particle size sand grains would be assumed (or the corresponding sand stone composed of

²Pressures can be used to uphold boundaries with our simulations: Instead of specifying walls non-slip and no-inflow conditions, it is possible to set the pressure to the corresponding hydrostatic values so that the fluid boundary does not move to model e.g. a vessel.

similar sized grains), to simulate macroscopic volumes, for channels in rock realistic dynamics could be simulated.

Non-isothermal flow Non-isothermal flow can be realized for the simulation by simulating the heat equation on the same grid as the fluid: Additionally, for the heat flow in the granular space, the triangular mesh must be extended over the granular particles, which is easy, taking e.g. the center of mass of the polygons as center and triangulating the polygon towards the edges. Solving the heat equation (respectively, the diffusion equation) on a FEM-grid is a standard procedure. Additionally, the viscosity has to be defined as temperature dependent. The simplest approach would then be to solve the heat equation in one sub-step, update the element-point dependent viscosities and then compute the solution for the Navier–Stokes equation.

Three dimensions The biggest problem with three dimensions is the CPU-time, rather than the algorithm. Imagining that we have a two-dimensional domain with $100 \times 100 = 10^4$ triangles, for the corresponding three dimensional domain we would have $100 \times 100 = 10^6$ tetrahedra, two orders of magnitude more. In comparison, the additional necessary velocity degrees for the z -direction (a factor of 1.5) or the nodes which have to be added to a triangle to obtain tetrahedral elements (10 instead of 6, an additional factor of 1.7) are negligible. A three-dimensional DEM simulation has been available in the group [77] now for considerable time, the algorithmic problems are elsewhere. While the solution of the FEM-equations can proceed as in two dimensions, the three-dimensional pore space between three dimensional polyhedra is much more complicated than between two-dimensional polygons. Obtaining a three-dimensional mesh between three-dimensional polyhedra is the actual algorithmic challenge for such a simulation.

7.5 Future simulation topics

With some of the above mentioned less complex program modifications, there are some research topics which could be undertaken in the nearer future.

Earthquake dynamics with dry and fluid-filled granular shear layers

The behavior of shear zones which are filled with rubble and water, as in the case of earthquake faults in the ocean, is hardly understood. While there shear simulations for zones filled with granulate, where the shear

zone can expand and contract [78], there are no simulations where there is additionally incompressible fluid and the volume is held constant: That is the actual situation when land masses start moving in large earthquakes. Nevertheless, running the simulation with periodic boundary conditions once that modification is available would remove some artificial choices at least for the boundaries.

Fluidization in earthquakes For the simulation of the fluidization in earthquakes, the mesh coarsening will be a necessary requirement, so that additional buffer zones with fluid can be created without unduly inflating the computational effort.

Landslides For the simulation of landslides, a meaningful determination of the fluid zone inside the granular heap will be necessary. In that case, it will be possible to vary the pore space by changing the width of the “particle shadow” and to determine critical porosities, as well as mechanisms like shear-band formation under the weight of the accumulated fluid and the actual fluid flow in such shear bands.

Space-resolved turbulence For this, the field-dependent mesh refinement and the realizations of periodic boundary conditions will be necessary. Making use of the adaptive grid, it seems worthwhile to investigate the relative accuracy of results for turbulence with fixed and with adaptive grid: Would it be possible to beat the accuracy for fixed grid if the mesh can actually be adapted to the actual spatial distribution of the flow field?

Chapter 8

Summary

We have discussed a two-dimensional microscopic (fluid goes around the particles) simulation method for polygonal granular particles in an incompressible Newtonian fluid. Our approach uses only the finite elements (FEM) for the incompressible Newtonian fluid and the discrete elements (DEM) for the particles without the need of additional data structures for the boundaries. In the discrete element method, polygonal particles are used: The force is proportional to the area overlap. The equations of motion of the particles from the corresponding force laws (elastic contact force, dissipative and frictional forces) are solved via the backward differentiation formula of 2nd-order (BDF2). The implementation of the incompressible Navier–Stokes equations via the Galerkin finite element method (FEM) is formulated as differential algebraic equations (DAE) with the pressures as the Lagrange parameters. The time integration is again via the Backward-Difference Formula of second order (BDF2) while the resulting non-linear equations are solved with the Newton–Raphson methods. The grid is obtained for Taylor–Hood elements from Delaunay triangulations with additional post-processing via the relaxation algorithm. For the combination of particle and fluid phase, using the implicit scheme—BDF2 as the integrator for both phases turned out to be stable. Even with the dynamic remeshing approach, we are able to obtain reasonable and “stable” results without large fluctuation in the pressure with reasonable CPU times; the remeshing effort is computationally much cheaper than the solution of the equations. For verification of the DEM-FEM code, we computed the wall correction factor for a regular dodecagon and compared it with analytical and simulation reference data for circular cylinders. Depending on the radius definition, the deviation from the simulation data is smaller than the latter’s deviation from the analytical values. Any further quest for single-particle results would only to

“precision without accuracy”: Realistic particles in granular materials in general are rough or have varying diameters and no symmetry, so the many-particle problem will be rather affected by the disorder (position of the particles and their variation in shape) than due to errors in the drag coefficient in the second digit.

We extend our fluid simulation method to free surfaces simulation via FEM: The motion of the surface elements is integrated out according to the velocity data obtained from the FEM-scheme on the surface. As we perform the time-integration of the FEM-code, the second-order Adams–Bashforth method turns out to be the most suitable integrator for the surface motion. We verified our free surfaces code by simulating the collapse of a water column. For the speed of the wavefronts, we get excellent agreement for large viscosity with the lubrication approximation. The agreement of the results with the experimental data for water is a further gratifying result. Compared to conventional efforts, which try e.g. to solve partial differential equations for the motion of the surface, the additional effort in our method with respect to new data structures, modeling assumptions etc. is negligible.

Using the developed code, we compare the compaction due to tapping in dry granulates to that with the system immersed in a viscous fluid to investigate the competition between a slowing-down of the dynamics due to the viscous forces of the fluid and the improved transmission of the tapping pulses through the fluid in the pore space. Our simulation has shown that the addition of fluid to a granular assembly can increase the sound velocity in the system, compared to the dry case. The high viscosity slowed down the compaction, irrespective whether the system was tapped only on the ground or on the whole boundary. The center of mass of a nearly square system drops faster for the dry case than for immersed particles. The granular column simulations show that for systems immersed under fluids, rolling of particles becomes less important than for the corresponding dry systems.

In this thesis, a working simulation code which can handle multiple polygonal particles in an incompressible Newtonian fluid has been developed. The current limitation of the our system size is due to the computation of the free fluid via FEM. Nevertheless, we have established a basis for future studies on problems which deal with the interaction between fluids and particles via the DEM-FEM approach. The limitations and possible improvements of the simulation are addressed in the next chapter.

Appendix A

Discretization of the Weak Form of the Navier–Stokes Equations

In this chapter, we show the derivation of the discretized weak form of the Navier–Stokes Eq. (3.33) and continuity Eq. (3.34) equations, namely the matrices M_α , K_α , etc. in Eq. (3.35) through Eq. (3.38) in detail. We start with the definitions of the approximated velocities, pressure and the corresponding test functions as the linear combination of the basis functions $\boldsymbol{\varphi}_{(x)}$, $\boldsymbol{\varphi}_{(y)}$, $\boldsymbol{\psi}$ in vector representation

$$\begin{aligned} u &= \boldsymbol{\varphi}_{(x)}^\top \mathbf{u}, & \phi^{(x)} &= \boldsymbol{\varphi}_{(x)}^\top \boldsymbol{\Phi}_{(x)}^*, \\ v &= \boldsymbol{\varphi}_{(y)}^\top \mathbf{v}, & \phi^{(y)} &= \boldsymbol{\varphi}_{(y)}^\top \boldsymbol{\Phi}_{(y)}^*, \\ P &= \boldsymbol{\psi}^\top \mathbf{P}, & \psi &= \boldsymbol{\psi}^\top \boldsymbol{\Psi}^*, \end{aligned}$$

where \mathbf{u} , \mathbf{v} , and \mathbf{P} are the nodal values of the velocities and pressures which are to be determined and $\boldsymbol{\Phi}_{(x)}^*$, $\boldsymbol{\Phi}_{(y)}^*$ and $\boldsymbol{\Psi}^*$ are the nodal values of the test functions. The weak form is given in Section 3.2.1 as

$$\begin{aligned} & \iint \phi^{(x)} \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} - X \right) - P \frac{\partial \phi^{(x)}}{\partial x} + \nu \left(\frac{\partial u}{\partial x} \frac{\partial \phi^{(x)}}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial \phi^{(x)}}{\partial y} \right) dx dy \\ &= \int_{\Gamma_u^N} \phi^{(x)} F_x d\Gamma, \end{aligned} \tag{3.24}$$

for the x -component of the Navier–Stokes equations and

$$\iint \psi \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) dx dy = 0, \tag{3.19}$$

for the continuity equation. The discretization can be done by substituting the approximated values u , v , P and the test functions $\phi^{(x)}$, $\phi^{(y)}$, ψ into Eq. (3.24) and Eq. (3.19) with the basic properties of the transpose operation of $\mathbf{a} = [a_x \ a_y]^\top$, $\mathbf{b} = [b_x \ b_y]^\top$ and $\mathbf{c} = [c_x \ c_y]^\top$

$$\begin{aligned}\mathbf{a}^\top \mathbf{b} &= \mathbf{b}^\top \mathbf{a}, \\ (\mathbf{a} + \mathbf{b})^\top &= \mathbf{a}^\top + \mathbf{b}^\top, \\ (\mathbf{ab})^\top &= \mathbf{b}^\top \mathbf{a}^\top, \\ \mathbf{a} (\mathbf{b}^\top \mathbf{c}) &= (\mathbf{ab}^\top) \mathbf{c},\end{aligned}$$

and the vector identity

$$(\mathbf{a} + \mathbf{b}) \cdot \mathbf{c} = \mathbf{a} \cdot \mathbf{c} + \mathbf{b} \cdot \mathbf{c},$$

in mind. Using the properties of the transpose we can first rewrite the test functions as

$$\begin{aligned}\phi^{(x)} &= \boldsymbol{\varphi}_{(x)}^\top \boldsymbol{\Phi}_{(x)}^* = (\boldsymbol{\Phi}_{(x)}^*)^\top \boldsymbol{\varphi}_{(x)}, \\ \phi^{(y)} &= \boldsymbol{\varphi}_{(y)}^\top \boldsymbol{\Phi}_{(y)}^* = (\boldsymbol{\Phi}_{(y)}^*)^\top \boldsymbol{\varphi}_{(y)}, \\ \psi &= \boldsymbol{\psi}^\top \boldsymbol{\Psi}^* = (\boldsymbol{\Psi}^*)^\top \boldsymbol{\psi},\end{aligned}$$

then continue with the discretization as the following:

Time derivative and the external force terms in Eq. (3.24) will give matrix M_1 and the first term in \mathbf{f}_1 :

$$\begin{aligned}& \iint \phi^{(x)} \left(\frac{\partial u}{\partial t} - X \right) dx dy \\&= \iint (\boldsymbol{\Phi}_{(x)}^*)^\top \boldsymbol{\varphi}_{(x)} \left(\frac{\partial \boldsymbol{\varphi}_{(x)}^\top \mathbf{u}}{\partial t} - X \right) dx dy, \\&= \iint (\boldsymbol{\Phi}_{(x)}^*)^\top \boldsymbol{\varphi}_{(x)} \left(\boldsymbol{\varphi}_{(x)}^\top \frac{\partial \mathbf{u}}{\partial t} - X \right) dx dy, \\&= (\boldsymbol{\Phi}_{(x)}^*)^\top \left[\underbrace{\iint \boldsymbol{\varphi}_{(x)} \boldsymbol{\varphi}_{(x)}^\top dx dy}_{M_1} \cdot \frac{\partial \mathbf{u}}{\partial t} - \underbrace{\iint \boldsymbol{\varphi}_{(x)} X dx dy}_{\text{First term in } \mathbf{f}_1} \right].\end{aligned}\tag{A.1}$$

Advection terms are discretized to $N_1(\mathbf{u})$:

$$\begin{aligned}
& \iint \phi^{(x)} \left(u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) dx dy \\
&= (\Phi_{(x)}^*)^\top \left[\iint \varphi_{(x)} \left(\varphi_{(x)}^\top \mathbf{u} \left(\frac{\partial \varphi_{(x)}^\top \mathbf{u}}{\partial x} \right) + \varphi_{(y)}^\top \mathbf{v} \left(\frac{\partial \varphi_{(x)}^\top \mathbf{u}}{\partial y} \right) \right) dx dy \right], \\
&= (\Phi_{(x)}^*)^\top \underbrace{\left[\iint \varphi_{(x)} \left(\varphi_{(x)}^\top \mathbf{u} \frac{\partial \varphi_{(x)}^\top}{\partial x} + \varphi_{(y)}^\top \mathbf{v} \frac{\partial \varphi_{(x)}^\top}{\partial y} \right) dx dy \cdot \mathbf{u} \right]}_{N_1(\mathbf{u})}. \tag{A.2}
\end{aligned}$$

Diffusion term is discretized to K_1 :

$$\begin{aligned}
& \iint \nu \left(\frac{\partial u}{\partial x} \frac{\partial \phi^{(x)}}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial \phi^{(x)}}{\partial y} \right) dx dy \\
&= \iint \nu \left[\left(\frac{\partial \varphi_{(x)}^\top \mathbf{u}}{\partial x} \right) \left(\frac{\partial (\Phi_{(x)}^*)^\top \varphi_{(x)}}{\partial x} \right) + \left(\frac{\partial \varphi_{(x)}^\top \mathbf{u}}{\partial y} \right) \left(\frac{\partial (\Phi_{(x)}^*)^\top \varphi_{(x)}}{\partial y} \right) \right] dx dy, \\
&= \iint \nu \left(\frac{\partial (\Phi_{(x)}^*)^\top \varphi_{(x)}}{\partial x} \frac{\partial \varphi_{(x)}^\top \mathbf{u}}{\partial x} + \frac{\partial (\Phi_{(x)}^*)^\top \varphi_{(x)}}{\partial y} \frac{\partial \varphi_{(x)}^\top \mathbf{u}}{\partial y} \right) dx dy, \\
&= (\Phi_{(x)}^*)^\top \underbrace{\left[\iint \nu \left(\frac{\partial \varphi_{(x)}}{\partial x} \frac{\partial \varphi_{(x)}^\top}{\partial x} + \frac{\partial \varphi_{(x)}}{\partial y} \frac{\partial \varphi_{(x)}^\top}{\partial y} \right) dx dy \cdot \mathbf{u} \right]}_{K_1}. \tag{A.3}
\end{aligned}$$

Pressure term to C_1 :

$$\begin{aligned}
& - \iint P \frac{\partial \phi^{(x)}}{\partial x} dx dy = - \iint \frac{\partial \phi^{(x)}}{\partial x} P dx dy, \\
&= - \iint \frac{\partial (\Phi_{(x)}^*)^\top \varphi_{(x)}}{\partial x} \psi^\top \mathbf{P} dx dy, \\
&= (\Phi_{(x)}^*)^\top \underbrace{\left[- \iint \frac{\partial \varphi_{(x)}}{\partial x} \psi^\top dx dy \right]}_{C_1} \mathbf{P}. \tag{A.4}
\end{aligned}$$

Neumann-type boundary term to \mathbf{f}_1 :

$$\int_{\Gamma_u^N} \phi^{(x)} F_x d\Gamma = \int_{\Gamma_u^N} (\Phi_{(x)}^*)^\top \varphi_{(x)} F_x d\Gamma = (\Phi_{(x)}^*)^\top \underbrace{\left[\int_{\Gamma_u^N} \varphi_{(x)} F_x d\Gamma \right]}_{\text{Part of } \mathbf{f}_1}. \tag{A.5}$$

Continuity equation :

$$\begin{aligned}
 & \iint \psi \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) dx dy = 0, \\
 & (\Psi^*)^\top \left[\iint \psi \left(\frac{\partial \varphi_{(x)}^\top \mathbf{u}}{\partial x} + \frac{\partial \varphi_{(y)}^\top \mathbf{v}}{\partial y} \right) dx dy \right] = 0, \\
 & (\Psi^*)^\top \left[\underbrace{- \iint \psi \frac{\partial \varphi_{(x)}^\top}{\partial x} dx dy \cdot \mathbf{u}}_{C_1^\top} - \underbrace{\iint \psi \frac{\partial \varphi_{(y)}^\top}{\partial y} dx dy \cdot \mathbf{v}}_{C_2^\top} \right] = 0. \quad (\text{A.6})
 \end{aligned}$$

Putting the resulting equations Eq. (A.1–A.6) leads us to the following

$$(\Phi_{(x)}^*)^\top \left[\int \dots \right] = 0, \quad (\Phi_{(y)}^*)^\top \left[\int \dots \right] = 0, \quad (\Psi^*)^\top \left[\int \dots \right] = 0,$$

where the nodal values $\Phi_{(x)}^*$, $\Phi_{(y)}^*$ and Ψ^* of the test functions can be placed outside of the brackets. Considering the arbitrary property of the nodal values¹ they will be dropped out from the resulting equations.

¹ $\Phi_{(x)}^* = \Phi_{(y)}^* = \Psi^* = \mathbf{0}$ tells us nothing about the equations.

Appendix B

Differential-Algebraic Equations

In this chapter we will discuss the formulation of differential-algebraic equations (DAE) by as example the simple pendulum. This allows us to discuss the particular properties of the DAE-formulation in comparison to approaches where the pressures are dealt with in a different way than as Lagrange multipliers, highlight the advantages and possible numerical problems. Considering a pendulum with mass m suspended from a string with length l from the origin in Fig.B.1. We start by introducing the Lagrange's equations of motion

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_k} \right) - \frac{\partial L}{\partial q_k} = 0, \quad k = 1, \dots, n, \quad (\text{B.1})$$

where q_1, \dots, q_n are the generalized position coordinates and $\dot{q}_1, \dots, \dot{q}_n$ are the generalized velocities. The Lagrangian L is defined with the kinetic energy T and the potential energy U

$$L = T - U. \quad (\text{B.2})$$

B.1 Polar Coordinate System

The common way to describe the simple pendulum problem (with a point-mass as bob) is by using the polar coordinate system (θ, r) , which we mention here for completeness sake. This approach gets around the problem of dealing with two dependent variables (x, y) by annihilating one variable via a coordinate transformation. The kinetic energy, potential energy and the Lagrangian can be

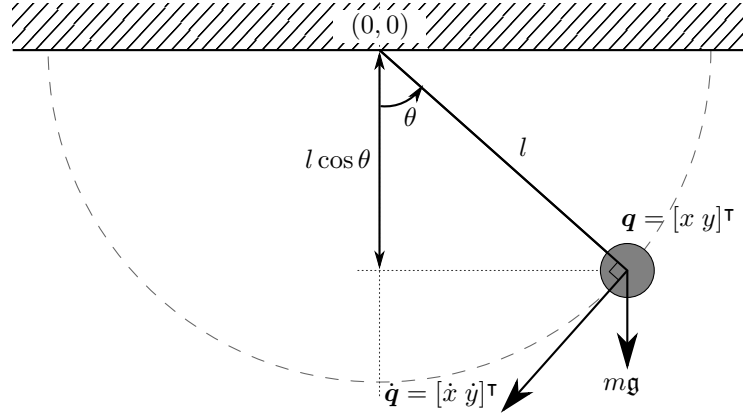


Fig. B.1 Simple pendulum problem with point mass m suspended from a string with length l from the origin.

written as

$$T = \frac{1}{2}m \left(\dot{\theta} l \right)^2, \quad (\text{B.3})$$

$$U = m\mathbf{g}l(1 - \cos \theta), \quad (\text{B.4})$$

$$L = \frac{1}{2}m\dot{\theta}^2 l^2 - m\mathbf{g}l + m\mathbf{g} \cos \theta, \quad (\text{B.5})$$

where \mathbf{g} denotes the gravitational acceleration and θ is the angle measured from the vertical (We will use the italic g , as is common in the field of DAE's, in various instances are the constraint function). Substituting L into the Lagrange's equations of motion Eq. (B.1) with $k = 1$ and $q_1 = \theta$, the equation of motion is obtained as a second order ordinary differential equation (ODE)

$$\ddot{\theta} l = -\mathbf{g} \sin \theta. \quad (\text{B.6})$$

An obvious disadvantage of the approach is that it depends on the geometry: For the simple geometric case of a constant radius, the approach with polar coordinates solves the problem automatically, while in the case where no appropriate coordinate system is available, coordinate transformations become more cumbersome. Another drawback in the given formulation with Lagrangian is that e.g. forces which are difficult to dealt with in a Lagrangian framework (friction) must be neglected in the derivation and be incorporated later into the derived equations: If such forces do not allow a formulation in polar coordinates, the approach collapses.

B.2 Cartesian Coordinate System with DAE

The pendulum can be dealt with directly in Cartesian coordinates (x, y) which are not independent of each other via Lagrange multipliers. To be consistent with the previous section, we outline again the approach via the Lagrangian, though the coefficients with multipliers can also be directly inserted into other equations of motion (in that case, obtaining the additional equations will need other, less formal arguments). We use the “Lagrange function”

$$L = T - U - \lambda_1 g_1(q) - \cdots - \lambda_m g_m(q) \quad (\text{B.7})$$

where $g_1(q) = 0, \dots, g_m(q) = 0$ are the constraint functions which describe the relations between the variables q_1, \dots, q_n , and λ_i are the Lagrange multipliers. When such constraints are introduced to a system, the number of the degrees of freedom is equal to the difference between number of variables and number of constraints. The constraint for our pendulum problem is that the distance l between the mass point and the origin is constant in the system

$$x^2 + y^2 - l^2 = 0. \quad (\text{B.8})$$

As we have two variables (x and y) and one constraint, the degrees of freedom will be 1 which is the same as in the polar coordinate system. Additional to the kinetic energy T and potential energy U we also need to define the constraint function g for the pendulum system as

$$T = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2), \quad (\text{B.9})$$

$$U = mgy, \quad (\text{B.10})$$

$$g = x^2 + y^2 - l^2 \quad (\text{B.11})$$

(It is common to write the constraint function g so that it is zero if the constraint is fulfilled: In that case, the deviation from zero immediately indicates the numerical error of the method). Inserting Eq. (B.9)–(B.11) into Eq. (B.7) leads to

$$L = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) - mgy - \lambda(x^2 + y^2 - l^2), \quad (\text{B.12})$$

so Eq. (B.1) with $\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial y}$, $\frac{\partial L}{\partial \lambda}$ becomes

$$m\ddot{x} = -2x\lambda, \quad (\text{B.13})$$

$$m\ddot{y} = -m\mathbf{g} - 2y\lambda, \quad (\text{B.14})$$

$$0 = x^2 + y^2 - l^2. \quad (\text{B.15})$$

The physical meaning of Lagrange multiplier λ in the pendulum problem is the tension in the string which maintains the mass point on the desired trajectory. Obviously, the constraint forces $-2\lambda x/m, -2\lambda y/m$ are in radial direction, as should be the case for a centripetal force, so that the purely formal arguments also lead to the intuitive physical result. While we have used only gravity as external force, we could add any imaginable force term in x - and y -direction, even impact-like forces: The Lagrange-multiplier formalism would also be applicable in this case, and impact like forces would be compensated by impact-like changes of λ .

The general form of a constrained mechanical system can be written in vector notation using generalized position coordinates $\mathbf{q} = [q_1 \dots q_n]^\top$, velocities $\mathbf{u} = [\dot{q}_1 \dots \dot{q}_n]^\top$ and $\boldsymbol{\lambda} = [\lambda_1 \dots \lambda_m]^\top$

$$\dot{\mathbf{q}} = \mathbf{u}, \quad (\text{B.16})$$

$$M(t, \mathbf{q})\dot{\mathbf{u}} = \mathbf{f}(t, \mathbf{u}, \mathbf{v}) - G^\top(t, \mathbf{q})\boldsymbol{\lambda}, \quad (\text{B.17})$$

$$0 = \mathbf{g}(t, \mathbf{q}), \quad (\text{B.18})$$

where \mathbf{f} are the applied forces (not including constraint forces) [22]. Using the subscript notation $h_{xy} = \frac{\partial}{\partial y} \left(\frac{\partial h}{\partial x} \right)$ for partial derivative, the positive definite generalized mass matrix M is defined as

$$M(\mathbf{q}) = T_{\dot{\mathbf{q}}\dot{\mathbf{q}}} = T_{\mathbf{u}\mathbf{u}} = \begin{bmatrix} T_{\dot{q}_1\dot{q}_1} & T_{\dot{q}_1\dot{q}_2} & \cdots & T_{\dot{q}_1\dot{q}_n} \\ T_{\dot{q}_2\dot{q}_1} & T_{\dot{q}_2\dot{q}_2} & \cdots & T_{\dot{q}_2\dot{q}_n} \\ \vdots & \vdots & \ddots & \vdots \\ T_{\dot{q}_n\dot{q}_1} & T_{\dot{q}_n\dot{q}_2} & \cdots & T_{\dot{q}_n\dot{q}_n} \end{bmatrix}, \quad (\text{B.19})$$

and the constraint Jacobian with the dimension of $(m \times n)$ as

$$G = \frac{\partial \mathbf{g}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial g_1}{\partial q_1} & \frac{\partial g_1}{\partial q_2} & \cdots & \frac{\partial g_1}{\partial q_n} \\ \frac{\partial g_2}{\partial q_1} & \frac{\partial g_2}{\partial q_2} & \cdots & \frac{\partial g_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial q_1} & \frac{\partial g_m}{\partial q_2} & \cdots & \frac{\partial g_m}{\partial q_n} \end{bmatrix}. \quad (\text{B.20})$$

By differentiating the constraints Eq. (B.18) with respect to time, the constraints at

velocity level is then

$$\frac{d\mathbf{g}}{dt} = \frac{\partial \mathbf{g}}{\partial \mathbf{q}} \frac{d\mathbf{q}}{dt} = G\mathbf{u}. \quad (\text{B.21})$$

For DAE's, various indices are defined, the higher the index, the higher the numerical complexity. The differentiation index 1 indicates that no differentiation is necessary, one necessary differentiation of the equations is “index 2”, and so on. Because differentiation “roughens” the solution (and increases the amplitude, as each time differentiation leads to an additional factor of $1/\tau$, with τ the time-step), the numerical implications of higher indices are immediately clear. Hence, the index 2 formulation (velocity level) of the differential-algebraic equations can be written as [17].

$$\dot{\mathbf{q}} = \mathbf{u}, \quad (\text{B.22})$$

$$M(t, \mathbf{q})\dot{\mathbf{u}} + G^\top(t, \mathbf{q})\boldsymbol{\lambda} = \mathbf{f}(t, \mathbf{u}, \mathbf{v}), \quad (\text{B.23})$$

$$G(t, \mathbf{q})\mathbf{u} = 0. \quad (\text{B.24})$$

For the simple pendulum problem we have

$$\overbrace{\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix}}^M \begin{Bmatrix} \ddot{x} \\ \ddot{y} \end{Bmatrix} + \overbrace{\begin{bmatrix} 2x \\ 2y \end{bmatrix}}^{G^\top} \lambda = \overbrace{\begin{Bmatrix} 0 \\ -m\mathbf{g} \end{Bmatrix}}^{\mathbf{f}}, \quad (\text{B.25})$$

$$\underbrace{\begin{bmatrix} 2x & 2y \end{bmatrix}}_G \begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix} = 0, \quad (\text{B.26})$$

where the time derivative of the constraint shows that the inner product between the position vector $[x \ y]^\top$ and the velocity vector $[\dot{x} \ \dot{y}]^\top$ is zero, i.e. the position vector (in radial direction) and the velocity vector (tangent to the circular trajectory) are orthogonal to each other.

Comparison with the Navier–Stokes equations in DAE-formulation Eq. (3.33) shows that the pressures indeed play the role of Lagrange multipliers. As the formulation with Lagrange-multipliers allowed also to deal with impact-like forces via fast (non-continuous) variation of λ , the DAE-formulation of the Navier–stokes equations allows an impact-like formulation for the pressures. The only limit in this case is the impact of the pressure variations on the velocities: If the variation of the velocities becomes unstable, the time integration will fail.

Appendix C

Using MEX-Files in MATLAB

With a built-in graphics library and huge collection of high-level numerical routines MATLAB is the programming language of choice for scientific computing. Its built-in toolboxes which work out of the box allow one to have an efficient way to develop the code and immediate inspection of the data through visualization without the need to go through the process of installing and loading the necessary modules. However, when dealing with huge amount of conditional statements in long iterations or multiple nested loops¹, one may hit a performance bottleneck with MATLAB script as the programming language itself is an interpreted language, i.e. the program lines are translated and executed line by line.

MATLAB allows the inclusion of subroutines from compiler languages such as C, C++, or Fortran; they can be called directly as if they were ordinary MATLAB functions. In this approach the C, C++, or Fortran code are compiled and made available as MATLAB executable (MEX). It enables the high performance of compiler languages while working within the MATLAB environment. In this chapter, it is shown how to rewrite a MATLAB function into a Fortran/MEX subroutine and the related solutions/workarounds for the problems which had to be faced during the development of the simulation are discussed.

C.1 Outer Product Computation of two Vectors

The subroutine for the Jacobian matrix in our FEM-code contains around 500 lines (Fortran/MEX-code), which is too long as an example. Instead, we show how to

¹An example is the constructing the Jacobian matrix J for the Newton–Raphson iterations in Eq. (3.88), where different if-conditions for element nodes for the boundary (and for different boundary conditions) and for nodes in the computation domains are incorporated.

write a Fortran/MEX subroutine for a function which computes the outer product of two vectors $\mathbf{a} = [a_1 \ a_2 \ \dots \ a_m]^\top$ and $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_n]^\top$

$$S = \mathbf{a} \cdot \mathbf{b}^\top = \begin{bmatrix} a_1 b_1 & a_1 b_2 & \cdots & a_1 b_n \\ a_2 b_1 & a_2 b_2 & \cdots & a_2 b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_m b_1 & a_m b_2 & \cdots & a_m b_n \end{bmatrix}. \quad (\text{C.1})$$

The implementation of the function in MATLAB can be written as the following:

```

1 function [sn, s, si, sj] = outerproduct_m(m, a, n, b)
2 % Computes the outer product of two vectors
3 % a (length = m) and b (length = n)
4 % and returns the resulting matrix as a sparse matrix:
5 % A vector s with the matrix entries and two index vector
6 % si and sj.
7
8 s = zeros(1, m*n);
9 si = zeros(1, m*n);
10 sj = zeros(1, m*n);
11 sn = 0;
12
13 for i = 1:m
14     for j = 1:n
15         sn = sn + 1;
16         s(sn) = a(i)*b(j);
17         si(sn) = i;
18         sj(sn) = j;
19     end
20 end
21
22 return

```

The function takes the vectors and their sizes as input arguments, and returns vectors **s**, **si**, and **sj** which are all the same length **sn**. Entry of vector **s(i)** contains the elements of matrix S located at the **si(i)**th row and the **sj(i)**th column in “sparse storage” form. This data structure is the same data structure that is used to construct the Jacobian matrix J for the Newton–Raphson iterations in (Eq. (3.88)), which is indeed a sparse matrix (with filling ratio below 0.3%).

C.2 Writing the Fortran/MEX-file

C.2.1 Preparation

The extension of the Fortran/MEX file should be the same as the usual respective compiler languages. That means for the Fortran-files, `.F` should be used (as for the `gfortran`-compiler). The file starts must begin with the inclusion of the Fortran header file `fintrf.h` which contains the declarations of the MATLAB (Application Program Interface) API function so that MATLAB is able to associate the Fortran-file-name under the respective MATLAB-function call. As in all higher programming languages, dummy names in the subroutines must not be the same as those for the function call in the calling program.

```
1  #include "fintrf.h"
```

Then we define the gateway routine `mexfunction()` as

```
3      subroutine mexFunction(nlhs, plhs, nrhs, prhs)
```

which serves as the entry point to the Fortran subroutine. The `mexfunction()` subroutine must be defined in all Fortran/MEX-file regardless of the intended function name. The arguments of the subroutine are the following

`nlhs` is the number of output arguments of the function.

`plhs` is an array containing the pointers to the output arguments.

`nrhs` is the number of input arguments of the function.

`prhs` is an array containing the pointers to the input arguments.

C.2.2 Declaring the Variables

```
5      implicit none
6      integer :: nlhs, nrhs
7      mwPointer :: plhs(*), prhs(*)
8      mwPointer :: mxGetPr
9      mwPointer :: mxCreateDoubleMatrix
10     integer*4, external :: mexPrintf
11     character*120 line
12     integer*4 :: line_out
```

The above declares arguments of the `mexfunction()` subroutine and the MATLAB API functions (`mxGetPr()`, `mxCreateDoubleMatrix()` and `mexPrintf()`) used in this MEX-file. The variables `line` and `line_out` are needed for displaying messages to the terminal.

Then the variables which stores the input/output data of MATLAB are declared with their corresponding pointers suffixed with `_ptr`. These variables will be used in the actual computation.

```

14      integer (kind=4), parameter :: maxn = 10000
15      integer (kind=4) :: m
16      real (kind=8) :: m_dble
17      real (kind=8), dimension(maxn) :: a
18      integer (kind=4) :: n
19      real (kind=8) :: n_dble
20      real (kind=8), dimension(maxn) :: b
21      integer (kind=4) :: sn
22      real (kind=8), dimension(maxn*maxn) :: s
23      real (kind=8), dimension(maxn*maxn) :: si
24      real (kind=8), dimension(maxn*maxn) :: sj
25
26      mwPointer :: m_ptr
27      mwPointer :: a_ptr
28      mwPointer :: n_ptr
29      mwPointer :: b_ptr
30      mwPointer :: sn_ptr
31      mwPointer :: s_ptr
32      mwPointer :: si_ptr
33      mwPointer :: sj_ptr

```

Notice that for variables of integer-type `m` and `n`, we have also declared their double precision counterparts (`_dble`). This is because MATLAB stores all numeric variables as double-precision floating-point values. The input data from MATLAB will be copied as double precision values into memory locations for the Fortran program then converted to integer type.

After the declaration part, one should check whether the correct input and output arguments are provided by the user during runtime. Passing the wrong number of arguments causes segmentation violation and crashes MATLAB without giving user any useful information for debugging. Instead of provoking such difficult to debug program crashed, understandable error message should be issues.

```

35      if ( nrhs .ne. 4 ) then
36          write(line, '("Four inputs required.\n")')
37          line_out = mexPrintf(line)

```



```

38         return
39     elseif ( nlhs .ne. 4 ) then
40         write(line, '("Four outputs required.\n")')
41         line_out = mexPrintf(line)
42         return
43     endif

```

C.2.3 Reading Input from MATLAB

The pointers which point to the memory locations of the input data from MATLAB are obtained from the function `mxGetPr()`.

```

45     m_ptr = mxGetPr( prhs(1) )
46     a_ptr = mxGetPr( prhs(2) )
47     n_ptr = mxGetPr( prhs(3) )
48     b_ptr = mxGetPr( prhs(4) )

```

Then, the input data (double precision type) are copied to the memory locations into Fortran variables using the `mxCopyPtrToReal8()` subroutine, where the first argument is the name of the pointer obtained from the previous step, the second argument is the Fortran variable to be copied into, and the third is the number of elements in the input data.

```

50     call mxCopyPtrToReal8(m_ptr, m_dble, 1)
51     m = int(m_dble)
52     if ( m .gt. maxn ) then
53         write(line, '("M (=",i0,") should be less than MAXN (=",
54 +             i0,")\n")') m, maxn
55         line_out = mexPrintf(line)
56         return
57     endif
58     call mxCopyPtrToReal8(a_ptr, a, m)
59     call mxCopyPtrToReal8(n_ptr, n_dble, 1)
60     n = int(n_dble)
61     if ( n .gt. maxn ) then
62         write(line, '("N(=",i0,") should be less than MAXN (=",
63 +             i0,")\n")') n, maxn
64         line_out = mexPrintf(line)
65         return
66     endif
67     call mxCopyPtrToReal8(b_ptr, b, n)

```

Variables `m_dble` and `n_dble` which hold the number of elements in each vectors are converted into integer type as mentioned in Section C.2.2. Then they are compared

with `maxn` to make sure that the input data from MATLAB are not larger than the variables `a` and `b` initialized in the declaration section of the code.

C.2.4 Actual Computation

The computation of the outer product comes after the copying of the necessary data from MATLAB. It is advantageous to perform the actual computation of the Fortran/MEX file in another subroutine, i.e. `outerproduct_f_comp()` in Section C.2.6. In that case, literally the same routine can be used in a Fortran-file for the development of the Fortran-routine and for the MEX-file, without tedious recopying of MEX-headers etc.

```

69      call outerproduct_f_comp(m, a(1:m), n, b(1:n),
70      +                          s(1:m*n), si(1:m*n), sj(1:m*n), sn)

```

C.2.5 Output to MATLAB

Next we discuss how the arguments are passed back to MATLAB. We start from the preparation of the output arguments `plhs()` using function `mxCreateDoubleMatrix(m,n,i)`. The function creates an `m`-by-`n` array in MATLAB and returns the resulting pointer to the created array. The last input argument of the function is the complex flag: 1 for imaginary data, 0 otherwise. Then, the data are copied into a MATLAB array pointed by the pointers e.g. `sn_ptr` using subroutine `mxCopyReal8ToPtr()`. Notice that in this step, we convert all non-double precision data to double precision via `DBLE()`.

```

72      plhs(1) = mxCreateDoubleMatrix(1, 1, 0)
73      plhs(2) = mxCreateDoubleMatrix(sn, 1, 0)
74      plhs(3) = mxCreateDoubleMatrix(sn, 1, 0)
75      plhs(4) = mxCreateDoubleMatrix(sn, 1, 0)
76      sn_ptr = mxGetPr(plhs(1))
77      s_ptr = mxGetPr(plhs(2))
78      si_ptr = mxGetPr(plhs(3))
79      sj_ptr = mxGetPr(plhs(4))
80      call mxCopyReal8ToPtr(DBLE(sn), sn_ptr, 1)
81      call mxCopyReal8ToPtr(s(1:sn), s_ptr, sn)
82      call mxCopyReal8ToPtr(DBLE(si(1:sn)), si_ptr, sn)
83      call mxCopyReal8ToPtr(DBLE(sj(1:sn)), sj_ptr, sn)

```

C.2.6 Actual Computation Subroutine

The actual computation of the outer product is as follows. The subroutine is similar to the MATLAB version except for the additional declarations for the variables.

```

90      subroutine outerproduct_f_comp(m, a, n, b, s, si, sj, sn)
91      implicit none
92      integer (kind=4), intent(in) :: m
93      real (kind=8), dimension(m), intent(in) :: a
94      integer (kind=4), intent(in) :: n
95      real (kind=8), dimension(n), intent(in) :: b
96      real (kind=8), dimension(m*n), intent(out) :: s
97      real (kind=8), dimension(m*n), intent(out) :: si
98      real (kind=8), dimension(m*n), intent(out) :: sj
99      integer (kind=4), intent(out) :: sn
100     integer (kind=4) :: i, j
101
102     sn = 0
103     do i = 1, m
104         do j = 1, n
105             sn = sn + 1;
106             s(sn) = a(i)*b(j);
107             si(sn) = i;
108             sj(sn) = j;
109         enddo
110     enddo
111
112     return
113 end subroutine outerproduct_f_comp

```

C.3 Compiling and Using Fortran/MEX-file

For our computation environment with Apple's OS X, the Fortran compilers which we found work well with the MEX-files are the following:

1. GNU Fortran (`gfortran`)'s binary from the Tools page at the R for Mac OS X developer's site.²
2. GNU Fortran from Homebrew³ which is an open source software package management system for OS X. The installation of `gfortran` can be done through: `$ brew install gcc`

²<http://r.research.att.com/tools/>

³<http://brew.sh/>

Other versions GNU Fortran compilers (such as the one from `hpc.sourceforge.net`) may lead to segmentation violations, program exceptions and other kinds of program crashes. To compile, the following command at the MATLAB terminal prompt must be executed:

```
>> mex outerproduct_f.F
```

MATLAB will autodetect the installed Fortran compiler. If the installed compiler cannot be found, one can copy `mexopts.sh` from `<path to matlab>/bin/` to the working directory and modify the file accordingly:

1. Provide the full path to `gfortran` in `FC`,
2. the full path to `libgfortran.dylib` in `FC_LIBDIR`,
3. and the full path to `libgfortranbegin.a` in `FC_LIBDIR2`.

These provide MATLAB the paths to the GNU Fortran libraries and executable installed in the system. To compile the source code with the modified option file,

```
>> mex -f mexopts.sh outerproduct_f.F
```

If the compilation is successful, it produces a compiled file with the extension `.mexmaci64` for 64-bit Intel-based Mac.

The MEX-file is called like an ordinary MATLAB's function, i.e. the name of the file without the extension is used, with arguments on the right-hand side of the function in round brackets and output arguments on the left-hand side (for multiple arguments, in angular brackets).

```
>> n = 1000
>> a = rand(1,n)
>> b = rand(1,n)
>> [snf, sf, sif, sjf] = outerproduct_m(n, a, n, b);
```

C.4 Remark

Fig. C.1 shows the time consumption of the MATLAB and Fortran/MEX functions for different vector size. One can see that if the size of the vector is not large enough, the difference in the computation time is not that grave. Creating a MEX-file can

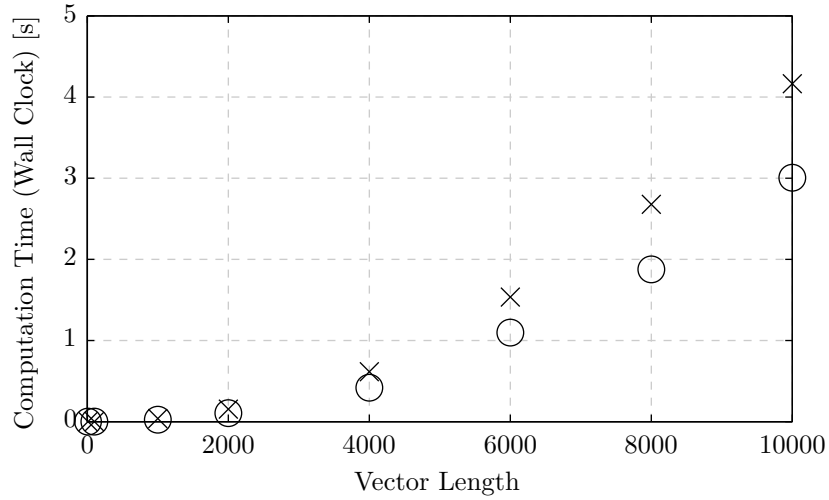


Fig. C.1 Time consumption of both MATLAB (×) and Fortran/MEX (○) version outer product functions for different vector sizes for MATLAB version R2014a on Mac OS X 10.10.

be more trouble than it is worth if the functions are not performance critical. For our example of an outer product, the Fortran/MEX version of the same function needs around 80 lines of additional code to pass the arguments between MATLAB and Fortran. Not to mention that slight mistakes in handling the data (e.g. wrong data type, wrong array size, etc.) might crash MATLAB without giving any useful information for debugging. Therefore, one should start from optimizing the original MATLAB codes in a program and only convert the bottleneck MATLAB functions into MEX-files. Usually these are the functions which use a lot of `if`-conditions in `for`-loops such as the construction of the sparse Jacobian matrix in our FEM-code which make them difficult to vectorize in MATLAB for better performance. Since MATLAB version 6.5 it has been promised that the MATLAB interpreter will precompile complicated functions if no effective interpreter execution seems possible; nevertheless, for several of our performance critical functions, the execution time was still not comparable with that for the MEX-codes.

References

- [1] J. C. Martin and W. J. Moyce. Part IV. An experimental study of the collapse of liquid columns on a rigid horizontal plane. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 244(882):312–324, 1952.
- [2] Gerald H. Ristow. Wall correction factor for sinking cylinders in fluids. *Physical Review E*, 55:2808–2813, March 1997.
- [3] MATLAB version 7.8.0.347 (r2009a), 2009.
- [4] Kai Höfler and Stefan Schwarzer. Navier–Stokes simulation with constraint forces: Finite-difference method for particle-laden flows and complex geometries. *Phys. Rev. E*, 61(6):7146–7160, Jun 2000.
- [5] Salah A. M. El Shourbagy, Shiro Okeda, and Hans-Georg Matuttis. Acoustic of sound propagation in granular materials in one, two, and three dimensions. *Journal of the Physical Society of Japan*, 77(3):034606, 2008.
- [6] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.
- [7] Martin Robinson, Marco Ramaioli, and Stefan Luding. Fluid-particle flow simulations using two-way-coupled mesoscale SPH-DEM and validation. *International Journal of Multiphase Flow*, 59(0):121–134, 2014.
- [8] S. Koshizuka and Y. Oka. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear Science Engineering*, 123(3):421–434, 1996.
- [9] O. Behrend. Solid-fluid boundaries in particle suspension simulations via the lattice Boltzmann method. *Phys. Rev. E*, 52:1164–1175, 1995.

- [10] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier–Stokes equation. *Physical Review Letters*, 56(14):1505–1508, 1986.
- [11] Yingkang Pan, Toshitsugu Tanaka, and Yutaka Tsuji. Direct numerical simulation of particle-laden rotating turbulent channel flow. *Physics of Fluids*, 13(8):2320–2337, 2001.
- [12] B. P. Leonard. A survey of finite differences of opinion on numerical muddling of the incompressible defective convection equation. In T. J. R. Hughes, editor, *Finite element methods for convection dominated flows; Proceedings of the Winter Annual Meeting, New York, N.Y., December 2–7, 1979*, volume 34 of *Applied Mechanics Symposia Series. AMD*, pages 1–17. American Society of Mechanical Engineers, 1979.
- [13] Takeo Kajishima. Numerical investigation of multiscale interaction between turbulent flow and solid particles (in japanese). In *54th National Congress of Theoretical and Applied Mechanics, Japan*, pages 217–218, 2005.
- [14] STORM/CFD2000 Version 5.0, 2003.
- [15] D. Z. Zhang and A. Prosperetti. Averaged equations for inviscid disperse two-phase flow. *Journal of Fluid Mechanics*, 267:185–219, 1994.
- [16] A. A. Johnson and T. E. Tezduyar. Simulation of multiple spheres falling in a liquid-filled tube. *Computer Methods in Applied Mechanics and Engineering*, 134:351–373, 1995.
- [17] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer, 2nd revised edition, 1996.
- [18] Kenneth Langstreth Johnson. *Contact Mechanics*. Cambridge University Press, 1987.
- [19] Hans-Georg Matuttis and Jian Chen. *Understanding the Discrete Element Method: Simulation of Non-Spherical Particles for Granular and Multi-body Systems*. John Wiley & Sons, 2014.
- [20] Wei Shen Cheng, Jian Chen, and Hans-Georg Matuttis. Granular acoustics of polyhedral particles. *AIP Conference Proceedings*, 1542(1):567–570, 2013.
- [21] P. A. Cundall and O. D. L. Strack. A discrete numerical model for granular assemblies. *Géotechnique*, 29(1):47–65, March 1979.

- [22] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1998.
- [23] C. William Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice Hall, 1971.
- [24] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.
- [25] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford Science Publications. Clarendon Press, 1996.
- [26] Arnold Nordsieck. On numerical integration of ordinary differential equations. *Mathematics of Computation*, 16:22–49, 1962.
- [27] L. W. B. Browne. The marker and cell technique. In John Noye, editor, *Numerical simulation of fluid motion: proceedings of an International Conference on the Numerical Simulation of Fluid Dynamic Systems held at Monash University, Melbourne*, pages 223–247. North-Holland Publishing Company, 1978.
- [28] Philip M. Gresho and Robert L. Sani. *Incompressible Flow and the Finite Element Method Volume 2: Isothermal Laminar Flow*. John Wiley and Sons, Ltd., 2005.
- [29] M. Bercovier and O. Pironneau. Error estimates for finite element method solution of the stokes problem in the primitive variables. *Numerische Mathematik*, 33(2):211–224, 1979.
- [30] C. Taylor and P. Hood. A numerical solution of the Navier–Stokes equations using the finite element technique. *Computers & Fluids*, 1(1):73–100, 1973.
- [31] Jerome J. Connor and Carlos Alberto Brebbia. *Finite element techniques for fluid flow*. Newnes-Butterworths, 1976.
- [32] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, 2nd revised edition, 1996.
- [33] W. E. Schiesser. *The numerical method of lines: Integration of partial differential equations*. Academic Press, 1991.

- [34] Garold J. Borse. *Numerical methods with MATLAB: a resource for scientists and engineers*. International Thomson Publishing, 1997.
- [35] Timothy A. Davis. UMFPACK: unsymmetric multifrontal sparse LU factorization package. Last visited 29th of September 2014.
- [36] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing (SISC)*, 7(3):856–869, 1986.
- [37] H.A. van der Vorst. BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific Computing (SISC)*, 13(2):631–644, 1992.
- [38] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, pages 157–172. ACM, 1969.
- [39] Jos van Kan, Guus Segal, and Fred Vermolen. *Numerical Methods in Scientific Computing*. VSSD, 2005.
- [40] Shi Han Ng and Hans-Georg Matuttis. Adaptive mesh generation for two-dimensional simulation of polygonal particles in fluid. *Theoretical and Applied Mechanics Japan*, 59:323–333, 2011.
- [41] Alejandro L. Garcia. *Numerical Methods for Physics*. Prentice Hall, 1994.
- [42] R.P. Bhatia and K.L. Lawrence. Two-dimensional finite element mesh generation based on stripwise automatic triangulation. *Computers & Structures*, 36(2):309–319, 1990.
- [43] Randolph E. Bank and Jinchao Xu. An algorithm for coarsening unstructured meshes. *Numerische Mathematik*, 73(1):1–36, 1996.
- [44] Shi Han Ng and Hans-Georg Matuttis. Simulating free boundaries in 2d fem flow simulation with direct time integration of the surface velocities. *Journal of Algorithms & Computational Technology*, 7(3):287–300, 2013.
- [45] Herbert E. Huppert. The propagation of two-dimensional and axisymmetric viscous gravity currents over a rigid horizontal surface. *Journal of Fluid Mechanics*, 121:43–58, 1982.

- [46] Francis H. Harlow and J. Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.
- [47] Stanley Osher and Ronald P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Number 153 in Applied Mathematical Sciences. Springer-Verlag, 2003.
- [48] Alexandre Caboussat, Vincent Maronnier, Marco Picasso, and Jacques Rappaz. Numerical simulation of three dimensional free surface flows with bubbles. In Eberhard Bänsch, editor, *Challenges in Scientific Computing - CISC 2002*, volume 35 of *Lecture Notes in Computational Science and Engineering*, pages 69–86. Springer Berlin Heidelberg, 2003.
- [49] Shi Han Ng and Hans-Georg Matthies. Simulation of flow problems with free boundaries in 2d using the finite element method. In *24th Computational Fluid Dynamics Symposium*, pages E3–4, 1–8. The Japan Society of Fluid Mechanics, 2010.
- [50] Julio Gratton, Swadesh M. Majaian, and Fernando Minotti. Non Newtonian gravity creeping flow. Technical report, International Centre for Theoretical Physics, Trieste (Italy), 1988.
- [51] Ronald L. Panton. *Incompressible Flow*. John Wiley & Sons, 2nd edition, 1996.
- [52] Robert Boucher Yasuki Nakayama. *Introduction to Fluid Mechanics*. Butterworth-Heinemann, 1999.
- [53] A. Ben Richou, A. Ambari, M. Lebey, and J.K. Naciri. Drag force on a circular cylinder midway between two parallel plates at $Re \ll 1$ part 2: moving uniformly (numerical and experimental). *Chemical Engineering Science*, 60(10):2535–2543, 2005.
- [54] James B. Knight, Christopher G. Fandrich, Chun Ning Lau, Heinrich M. Jaeger, and Sidney R. Nagel. Density relaxation in a vibrated granular material. *Physical Review E*, 51:3957–3963, May 1995.
- [55] E. Ben-Naim, J.B. Knight, E.R. Nowak, H.M. Jaeger, and S.R. Nagel. Slow relaxation in granular compaction. *Physica D: Nonlinear Phenomena*, 123(1-4):380–385, 1998.
- [56] E. R. Nowak, J. B. Knight, M. L. Povinelli, H. M. Jaeger, and S. R. Nagel. Reversibility and irreversibility in the packing of vibrated granular material. *Powder Technology*, 94(1):79–83, 1997.

- [57] P. Philippe and D. Bideau. Numerical model for granular compaction under vertical tapping. *Physical Review E*, 63:051304, April 2001.
- [58] P. Philippe and D. Bideau. Compaction dynamics of a granular medium under vertical tapping. *EPL (Europhysics Letters)*, 60(5):677, 2002.
- [59] P. Philippe and D. Bideau. Granular medium under vertical tapping: Change of compaction and convection dynamics around the Liftoff threshold. *Physical Review Letters*, 91:104302, September 2003.
- [60] Ph. Ribière, P. Richard, P. Philippe, D. Bideau, and R. Delannay. On the existence of stationary states during granular compaction. *The European Physical Journal E*, 22:249–253, 2007.
- [61] J. A. Dijksman and M. van Hecke. The role of tap duration for the steady-state density of vibrated granular media. *EPL (Europhysics Letters)*, 88(4):44001, 2009.
- [62] G. Lumay and N. Vandewalle. Compaction of anisotropic granular materials: Experiments and simulations. *Physical Review E*, 70:051314, November 2004.
- [63] Emanuele Caglioti, Vittorio Loreto, Hans J. Herrmann, and Mario Nicodemi. A “tetris-like” model for the compaction of dry granular media. *Physical Review Letters*, 79:1575–1578, August 1997.
- [64] J.-M. Hertzsch. Cellular model for the compaction of a vertically tapped granular column. *The European Physical Journal B - Condensed Matter and Complex Systems*, 18:459–466, 2000.
- [65] V. Ratnaswamy, A.D. Rosato, D. Blackmore, X. Tricoche, N. Ching, and L. Zuo. Evolution of solids fraction surfaces in tapping: simulation and dynamical systems analysis. *Granular Matter*, 14:163–168, 2012.
- [66] T. Boutreux and P. G. de Gennes. Surface flows of granular mixtures: I. general principles and minimal model. *J. Phys. I France*, 6(10):1295–1304, 1996.
- [67] Tetsuo Akiyama. personal communication, 2001.
- [68] L. Rondon, O. Pouliquen, and P. Aussillous. Granular collapse in a fluid: Role of the initial volume fraction. *Physics of Fluids*, 23(7):073301, 2011.
- [69] MUMPS: a parallel sparse direct solver. Last visited 29th of December 2014.
- [70] PARDISO 5.0.0 solver project. Last visited 29th of September 2014.

- [71] Gerd Gudehus. *Physical Soil Mechanics*. Springer, 2010.
- [72] Salah A. M. El Shourbagy, Shinichi Morita, and Hans-Georg Matuttis. Simulation of the dependence of the bulk-stress-strain relations of granular materials on the particle shape. *Journal of the Physical Society of Japan*, 75(10):104602, 2006.
- [73] C. Lubich. On projected Runge-Kutta methods for differential-algebraic equations. *BIT Numerical Mathematics*, 31(3):545–550, 1991.
- [74] Hirotaka Sakaue, Katsuaki Morita, Mineo Goto, Megumi Shimizu, Tsuyoshi Hyakutake, and Hiroyuki Nishide. *Chemical Flow Control Method Using Combination of Hydrophobic and Hydrophilic Coatings*. American Institute of Aeronautics and Astronautics, 2008.
- [75] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters, 1996.
- [76] Salah A. M. El Shourbagy and Hans-Georg Matuttis. Dependence of the angle of repose of heaps on the particle shape. *RIMS Kokyuroku: Mathematical Aspects of Pattern Formation in Complex Fluid*, (1413):75–83, 2005.
- [77] Jian Chen. *Discrete element method for 3D simulations of mechanical systems of non-spherical granular materials*. PhD thesis, The University of Electro-Communications, 2012.
- [78] Hans-Jürgen Tillemans and Hans J. Herrmann. Simulating deformations of granular solids under shear. *Physica A: Statistical Mechanics and its Applications*, 217(3-4):261–288, 1995.

List of Publications Related to the Thesis

1. Shi Han Ng and Hans-Georg Matuttis, “Adaptive mesh generation for two-dimensional simulation of polygonal particles in fluid”, *Theoretical and Applied Mechanics Japan*, 2010, Vol. 59, pp. 323–333. (Related to Chapter 3)
2. Shi Han Ng and Hans-Georg Matuttis, “Two-dimensional microscopic simulation of granular particles in fluid”, *Theoretical and Applied Mechanics Japan*, 2011, Vol. 60, pp. 105–115. (Related to Chapter 5)
3. Shi Han Ng and Hans-Georg Matuttis, “Simulating Free Boundaries in 2D FEM Flow Simulation with Direct Time Integration of the Surface Velocities”, *Journal of Algorithms & Computational Technology*, 2013, Vol. 7 No. 3, pp. 287–300. (Related to Chapter 4)
4. Shi Han Ng and Hans-Georg Matuttis, “Effect of the surrounding fluid on the compaction of granular materials by tapping: Slow dynamics made slower?”, *AIP Conference Proceedings*, 2013, Vol. 1518 No. 1, pp. 372–378. (Related to Chapter 6)
5. Shi Han Ng and Hans-Georg Matuttis, “Polygonal particles in fluids”, *AIP Conference Proceedings*, 2013, Vol. 1542 No. 1, pp. 1138–1141. (Related to Chapter 6)

Author Biography

NG SHI HAN (ウン シ ハン) 国籍：マレーシア 1985 年 4 月 24 日 生

2005 年 9 月	奈良工業高等専門学校	電子制御工学科	編入学
2008 年 3 月	奈良工業高等専門学校	電子制御工学科	卒業
2008 年 4 月	電気通信大学	電気通信学部 知能機械工学科	編入学
2010 年 3 月	電気通信大学	電気通信学部 知能機械工学科	卒業
2010 年 4 月	電気通信大学	情報理工学研究科	
		知能機械工学専攻 博士前期課程	入学
2012 年 3 月	電気通信大学	情報理工学研究科	
		知能機械工学専攻 博士前期課程	修了
2012 年 4 月	電気通信大学	情報理工学研究科	
		知能機械工学専攻 博士後期課程	入学

Acknowledgment

I would like to express my sincere gratitude to my advisor and mentor Associate Professor Hans-Georg Matuttis for his continuous support and patient guidance of my study and research since the first day I joined his group as final-year undergraduate student. I admire his vast knowledge in many areas (physics, computational physics, programming skill, etc.) and his inspiration, enthusiasm in teaching. I could not have imagined having a better advisor and mentor for my study.

I would also like to thank the rest of the supervisory committee: Prof. Takeshi Miyazaki, Prof. Hiroshi Maekawa, Prof. Tomio Okawa and Prof. Naruo Sasaki for their encouragement and insightful comments.

Finally, I would also like to thank my family for the unconditional supports, the Malaysian Public Service Department (JPA) and Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) for the scholarships and the opportunity of studying in Japan.