Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2021, 9, 1257-1270
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Improved Grid Relaxation with Zero-Order Integrators

Jan Mueller, Hans-Georg Matuttis

Department of Mechanical and Intelligent Systems Engineering, The University of Electro-Communications, Chofu, Japan
Email: jan@matuttis.mce.uec.ac.jp

How to cite this paper: Mueller, J. and Matuttis, H.-G. (2021) Improved Grid Relaxation with Zero-Order Integrators. *Journal of Applied Mathematics and Physics*, **9**, 1257-1270.
https://doi.org/10.4236/jamp.2021.96086

Received: April 30, 2021
Accepted: June 14, 2021
Published: June 17, 2021

Abstract

In this research, we have improved a relaxation method for triangular meshes intended for finite element fluid simulations which contain discrete element particles. The triangle edges are treated as springs which relax their lengths towards a "better" force equilibrium where the triangles are closer to equilateral shape. The actual kernel is an improved zero order integrator which is able to follow reconfigurations of the particles faster than earlier methods. The improved relaxation allows larger timesteps in the flow simulation and leads to more stable, faster mesh reconfigurations for fast moving particles in the flow. Additionally, this demonstrates how integrators of the same order zero can nevertheless have different convergence speeds towards equilibrium.

Keywords

Zero-Order Solver, Equilibration, Grid Generation, Fluid Mechanics

1. Introduction

Systems of particles in fluids exhibit a wide range of phenomena like sedimentation, erosion, liquefaction as well as landslides, which go far beyond the rheology of homogeneous fluids. For the corresponding simulations, carelessness in the implementation of the boundary conditions of the particles introduces noise and inaccuracies in the interaction computation between particles and fluid and may annihilate the validity of a simulation method altogether. As the most reliable approach to take into account both the non-sphericity of granular particles, as well as the geometry of the pore space between those particles, we have devised a two-dimensional simulation that combines a discrete element method (DEM) of polygonal granular particles [1] with a finite element method (FEM) fluid on a triangular grid [2]. While the mesh generation itself is performed via Delaunay triangulation and discretizes the pore space between the particles exactly, the

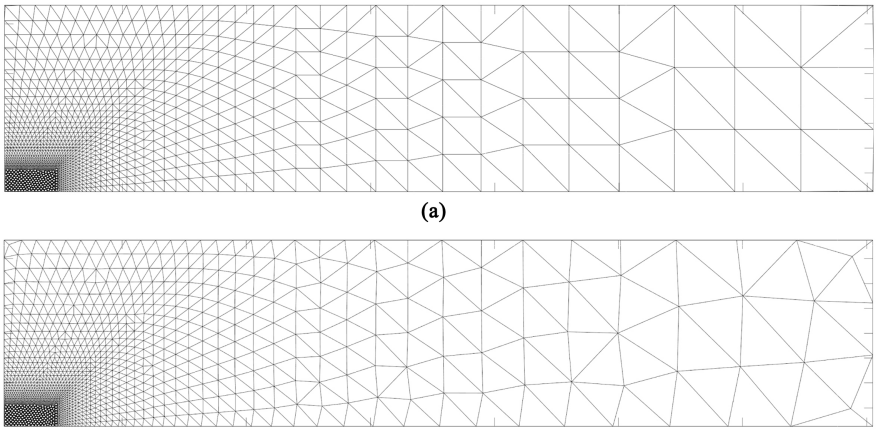
shape of the resulting triangles is not necessarily computationally ideal. It is advisable to improve the resulting mesh by moving the grid points inside of the fluid according to criteria which will be explained in Section 1.1. For this purpose, in the grid relaxation implemented according to Ng *et al.* [3], grid points are treated as connection points between springs which are placed along the edges of the FEM-triangles. Those springs are then relaxed towards equilibrium lengths of corresponding equilateral triangles by employing a zero-order ODE-solver. The principle is explained in Section 2 for chains of one-dimensional springs, while the actual implementation for the two-dimensional flow simulation is discussed in Section 3. Only recently, we found that Persson and Strang [4] had earlier proposed a similar relaxation approach, but to obtain a grid which was as balanced as possible over the whole simulation domain. In contrast, our relaxation is implemented to improve the grid quality locally, to retain fine meshes near the particle boundaries and coarse meshes towards the far away buffer zones, while still reshaping the mesh elements into a more equilateral (and thus more favourable) shape. As it is not the dynamics of the spring system but the final equilibrium positions which are of interest, any spring oscillations that occur on the way to this equilibrium can be "cut off" by zero-order integrators. This approach is simpler than an implementation of a non-linear solver, which would have to take into account position changes of the grid points, any "non-linear" local elasticities due to direction, connectivity as well as actual and equilibrium lengths of the springs. Nevertheless, as we will show, zero-order integrators may still have different convergence speeds towards equilibrium.

Unfortunately, the relaxation method we had used previously [3] (derived by introducing a first-order error into a second-order integrator) needed rather many timesteps to reach equilibrium, which made it computationally not very efficient. Ongoing relaxation during successive physical timesteps leads to changes of the triangle areas which introduced noise, to which the step adaption control of the integrator reacted by reducing the timestep. For small triangles, oscillations in the mesh could be introduced, which destabilized the flow simulation [5].

In this article, we want to modify the integrator so that it is still of zeroth-order but adapts faster to the reconfiguration of the particle positions to speed up the relaxation, increase the size of the timestep and avoid any oscillations in the mesh for small triangles.

1.1. Meshing Criteria for Particles in Fluids

Simulations of particles in fluids usually require a variety of regions with different mesh sizes which must be joined together (see **Figure 1**). A fine mesh around the particles and a coarse mesh far away from any regions of interest may be used in conjunction. Algorithms are needed to ensure that the mesh size in the border regions between meshes does not change too abruptly, which would lead to unsteady flow behavior due to fast changes in the wave resistance and therefore to artifacts in the propagation of the FEM-solution. Further, degenerate



(a)

(b)

Figure 1. Typical configuration with fine mesh around the particles (bottom left of the computational domain), coarse mesh far away from the region of interest (right boundary) and graded mesh in between. In (a), the initial grid from the Delaunay triangulation is shown, and in (b), the same grid after 60 relaxation steps.

mesh shapes should be avoided, as they would lead to bad conditioning of the underlying system of equations. To improve the efficiency of the simulation, it is advisable to introduce graded meshes: In regions of interest, the size of the mesh elements needs to be kept small e.g. around the particles or where high gradients of the flow fields (pressure, velocity) need to be resolved with sufficient accuracy. On the other hand, many setups feature large buffer regions stretching towards the boundaries. Large mesh elements are sufficient there, because far away from the region of interest the gradients in the velocity and pressure fields are much smaller. When we then treat the meshes as springs acting between grid points, we can obtain close to equilateral triangles with a suitable force model: The equilibrium configuration can be obtained by calculating the spring forces based on the current positions of the mesh points and "integrating" them with a zero-order algorithm which will return the positions of the mesh points in the force equilibrium. As we will show here, an adept selection of this zero-order integrator leads to noticeable improvements in the relaxation behaviour. For the discussion of the topic, we will restrict ourselves to the two-dimensional case, but the situation in three dimensions is analogous.

1.2. Triangular Grids for Particles in Fluids

The forces of a fluid acting on a particle are obtained from the boundary integral, in two dimensions along the outline Γ, as

$$\boldsymbol{F}^{\text{Drag}} = \int_{\Gamma} \left\{ -p\delta_{ij} + \eta\left((\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^{\mathrm{T}} \right) \right\} \cdot \hat{n}\, \mathrm{d}l \tag{1}$$

from the pressure p (form drag), as well as from the velocity field $\boldsymbol{u}$ and the viscosity η (friction drag). In turn, the force of the particle acting on the fluid is exerted as a boundary condition of the fluid domain. This means that any simulation approach that requires a physical fluid solid interaction becomes invalid

when the length of the boundary is discretized via an uncontrolled approximation. For structured square grids, this is indeed the case and it may come as a surprise that mesh refinement does not remedy this issue. As Figure 2 shows, the closest approximation of a unit circle on a square grid always has a "circumference" (sum over the outlining element edges between the gridpoints) of 4, even for successive refinements: It is not possible to approach the correct circumference of 2π, although the approximation of the area improves.

When we consider not only a single particle, but the very complicated pore space between sedimenting (or sedimented) particles, it is clear that a triangulation is able to give a much better approximation than a decomposition into squares. In particular, for Equation (1), square grids will always result only in zero-order approximations, while for polygonal particles; triangular grids can represent the outline exactly, without any need for approximation.

2. Relaxation Algorithm

For a given setup of mesh points (from the outlines of the particles, as well as existing stencil points in the fluid space) we can always create a space filling triangular mesh via Delaunay triangulation. More precisely, it is a so-called "constrained" Delaunay triangulation, which may violate the property that no points are contained in the circumcircle of any triangle, which is the condition for Delaunay triangulations in the strict sense. Nevertheless, the arbitrary particle positions (we assume polygonal particles throughout the paper) result in arbitrary positions of the grid points and in turn to mesh triangles with arbitrary angles. This has to be taken into account, as the quality of the mesh has to be maintained so that FEM-triangles do not contain angles above 135 degrees [6]. The closer the triangles are kept to an equilateral shape, the better the condition number of the Jacobian of the FEM-system becomes.

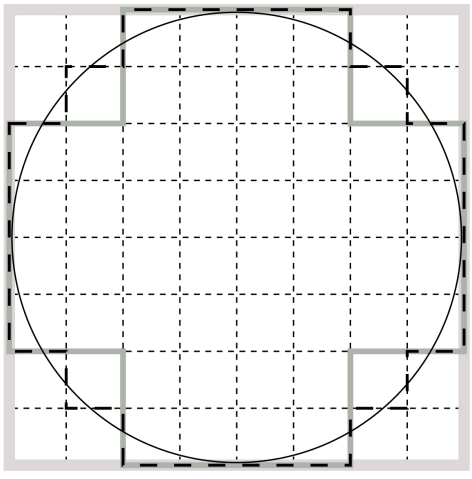


Figure 2. Refinement of a structured square mesh around a unit circle: Coarse mesh (thick grey line), intermediate mesh (thin grey line), fine mesh (dashed black line). No matter how fine the square mesh is, the circle will always be represented with a circumference of 4 instead of 2π.

2.1. Relaxation Method

The relaxation method in ref. [3] was derived from a Störmer-Verlet-integrator of second order

$$r_{n+1} = 2r_n - r_{n-1} + \iota^2 a_n \tag{2}$$

with the "virtual" relaxation timestep ι. We use Greek ι (iota), which looks like the symbol of the physical timestep τ with something missing, to express that ι is deficient in physical meaning and only a (dimensionless/virtual) iteration parameter. The acceleration a_n is obtained from the spring force, as deviation from the equilibrium length and can therefore be positive (under compression) or negative (under tension). From Equation (2) we obtain

$$r_{n+1} = r_n + r_n - r_{n-1} + \iota^2 a_n, \tag{3}$$

$$= r_n + \iota \underbrace{\frac{r_n - r_{n-1}}{\iota}}_{v_n} + \iota^2 a_n. \tag{4}$$

In [3], we dropped the term v_n, to introduce an error of first order with

$$r_{n+1} = r_n + \iota^2 a_n. \tag{5}$$

Accordingly, the remaining algorithm had an accuracy of order zero, which for the solution of Newton's equation of motion converges to the force-equilibrium. For grids with roughly uniform grid size, the convergence of this method is very satisfying. Nevertheless, for a grid with short and long edges, or for large spring constants k, convergence issues arise in the form of grid oscillations (between corner nodes of the same triangles, or neighbouring triangles) and in the form of mesh elements getting squeezed by larger neighbouring elements. As a remedy, in [5] we have introduced an additional spring force (modelled as acceleration)

$$a_{n,gs} = k \cdot \sum_i \operatorname{sgn}\left(A_{n,i} - A_n\right) \cdot \sqrt{\left|\left(A_{n,i} - A_n\right)\right|} \tag{6}$$

which uniformly wants to grow all edges of an element if its area A_n is smaller than that of its neighbours $A_{n,i}$ or shrink them if its area is larger instead. Grid oscillations, however, require a substantial reduction in the number of relaxation timesteps ι per physical timestep τ to be suppressed. Otherwise, they are pesky at least because they require a reduction of the physical timestep τ within the adaptive timestep control (see Section 4.2). In the worst case, they are disastrous, as the oscillations considerably change the triangle sizes from one physical timestep to the next, which leads to locking of the FEM due to oscillatory noise in the solution [5].

When we consider the zero-order approximation in Equation (5) above as one extreme of Equation (4), while dropping the acceleration term would lead to

$$r_{n+1} = r_n + \iota \underbrace{\frac{r_n - r_{n-1}}{\iota}}_{v_n}. \tag{7}$$

This does not contain the spring forces at all and would be quite useless on its own. Now we can look for a "better" algorithm in the convex hull of both Equa-

tion (5) and Equation (7) with a partitioning

$$r_{n+1} = r_n + \iota p v_n + \iota^2 (1-p) a_n, \tag{8}$$

$$= r_n + p(r_n - r_{n-1}) + \iota^2 (1-p) a_n, \tag{9}$$

which for a partition factor $0 < p < 1$ would still be of order zero and thus preserve the convergence.

2.2. Linear Chain to Determine the Parameters

The effects of the relaxation from Equation (9) can be visualized by implementing a one-dimensional linear chain that starts out with random segment lengths and then relaxes the configuration towards equal distances between nodes. The average segment length we use is $r_n - r_{n-1} = 1$. Partitioning values for $0.8 \le p \le 0.9$ show very good stability while the results still converge relatively fast: For values of $p > 0.9$, the convergence speed is slowed down (up to $p = 1$, where the position of the mesh nodes does not change at all). In case of $p = 0$, Equation (9) becomes identical to Equation (5) (our old algorithm), which we will use for comparison.

Figure 3 shows the linear chain with 20 mesh nodes, with the spring force

$$F^{ij} = k(x_j - x_i) \tag{10}$$

between the nodes, while the first and last node ($i = 1$ and $i = 20$) were fixed. The relaxation timestep is an arbitrary (within the limits of convergence), dimensionless constant and taken as $\iota = 0.1$. For partitioning values of $p = 0$ (old algorithm) and $p = 0.8$, we can see clear differences for $k = 50$: The old algorithm shows oscillations that subside only slowly, while the new algorithm

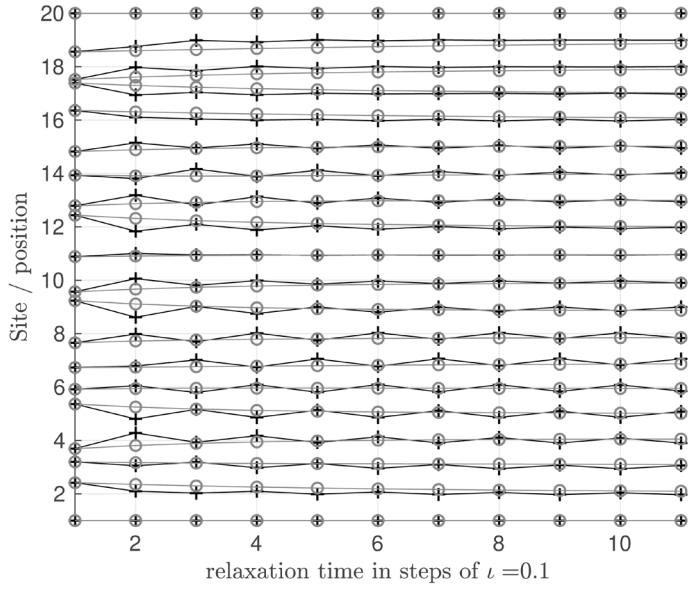


Figure 3. The old relaxation method (black lines with +, analogous to $p = 0$ in Equation (9)) shows various oscillations for $k = 50$ while the new relaxation method with $p = 0.8$ (grey lines with ∘) shows stable convergence.

ensures a smooth convergence towards the equilibrium points. With $p=0$ as in Equation (5), the oscillations around the equilibrium position will occur for $k \leq 30$ or even lower values in case of too large ι (or too large changes of r_n). In the language of numerical analysis, this would correspond to an ι which is larger than the stability radius of the integrator. The new algorithm Equation (9) with $p=0.8$ properly locates the equilibrium positions without these issues. This more stable "advection" towards the equilibrium position occurs because the "velocity part" and the "acceleration part" compensate each other in case of larger ι, while with the old algorithm Equation (5) the points would oscillate around their equilibrium positions in such a case. The new relaxation with $p=0.8$, shows no oscillations up to spring constants of $k=150$, whereas the "old" relaxation in Equation (5) already exhibits oscillations which diverge towards infinity with these settings. In terms of stability, the possibility to equilibrate systems without oscillations for spring constants up to $k=150$ for $p=0.8$ instead of $k=30$ for $p=0$ (the old algorithm) corresponds to at least a factor of 150/30 = 5 in the grid size (towards smaller grids), or conversely for the same grid size translates into a convergence speed which is faster by a factor of 150/30 = 5.

While in the theory of differential equations it is customary to compute convergence radii, this cannot be done in our case, because we do not solve the underlying differential equation in detail, but only compute suitable equilibrium solutions: The improvements are shown by the numerical results, not by any algebraic relations.

2.3. Failure of an "Implicit Euler" Relaxation

The term $\iota p v_n$ in Equation (8), *i.e.* the use of the velocity of step n, corresponds to an "explicit Euler" step. For the linear chain in the previous Section 2.2, we also tried out the corresponding "implicit Euler" step: We extrapolated the velocity to timestep $n+1$ as

$$\tilde{v}_{n+1} = v_n + \iota a_n. \tag{11}$$

and tried to advance the integrator with the $\iota p \tilde{v}_{n+1}$ instead of $\iota p v_n$. It turned out that the convergence was worse and stability (absence of oscillations) was considerably reduced. The reason becomes obvious when one inserts Equation (11) into Equation (9) so that

$$r_{n+1} = r_n + p\left(r_n - r_{n-1}\right) + \iota^2 a_n, \tag{12}$$

which corresponds to the original Verlet-method in the form of Equation (3) with a prefactor p in front of $\left(r_n - r_{n-1}\right)$: The accuracy order of the "implicit" step thus becomes higher than the one of the "explicit" step in the zero-order approach of Equation (5) because an additional first order term is added, and oscillations get amplified instead of curtailed. Therefore, this "implicit Euler" approach is not suitable for equilibration, e.g. our grid relaxation, because the "advection" to the equilibrium position is disturbed by the unwanted noise of

the oscillations. That the improved accuracy delays the convergence to the equilibirum shows again that conventional stability theory cannot be meaningfully applied for zero-order equilibria.

3. Implementation in the Simulation

In our two-dimensional, combined DEM-FEM simulation [2], the mesh data are passed from the Delaunay triangulation routine (including node coordinates and a connectivity list) to the relaxation routine. This information is used to calculate the edge lengths and areas of all mesh elements and further to generate a list of neighbouring elements. By comparing the areas of neighbouring elements, we get the spring accelerations based on area differences $a_{n,gs}$ from Equation (6). To force the triangular elements into a more equilateral shape, we compare the length of each edge l_j to the average edge length l_{ave} of the associated element and determine the related spring accelerations as

$$a_{n,eq} = \left(l_{ave} - l_j\right) \cdot k \quad \text{with} \quad l_{ave} = \frac{1}{3}\sum_{j=1}^{3} l_j. \tag{13}$$

For each edge, the accelerations are then summed up into

$$a_n = a_{n,eq} + a_{n,gs} \tag{14}$$

and applied to the mesh nodes according to Equation (9). The nodes may move individually or in groups, in parallel or in opposite directions. Note that by treating directly the accelerations, not the forces, there is no need to additionally assign a virtual mass to the mesh nodes. In an additional buffer, the mesh node positions from the previous relaxation step are held to be used as r_{n-1} in the next timestep. For the initialization of the first step, $r_{n-1} = r_n$ is used (mesh nodes are at rest initially).

The relaxation process is performed on the existing Delaunay triangulation for every physical timestep with a set number of relaxation steps N_{short}. In regular intervalls (usually every 50 physical timesteps), the grid is newly triangulated and relaxed over an increased amount of steps N_{long}, with additional re-triangulations every N_{tri} steps.

4. Comparison of Relaxation Algorithms

To verify the effects of the new relaxation algorithm, we keep spring constant $k = 1$ and step size $\iota = 0.05$ from the old algorithm based on Equation (5) and set the partitioning to $p = 0.9$. Only the number of relaxation steps per physical timestep will be adjusted.

4.1. Effect on the Mesh Geometry

First, we compared the behaviour of the old and new relaxation algorithms for the graded mesh around a granular step. Here, the two-dimensional behaviour is very similar to the one-dimensional case discussed in Section 2.2. If both algorithms are set to $N_{short} = 4$, $N_{long} = 8$ and $N_{tri} = 4$, the new algorithm is

slower to converge towards an equilibrium configuration. In Figure 4(c) this can easily be observed, as the effect is most pronounced in the coarse regions of the mesh: For the same number of relaxation steps, in the mesh relaxed with the new algorithm the triangles are less equilateral compared to their counterparts that were relaxed using the old algorithm. This means more relaxation steps would be required to reach the same mesh quality. For the fine mesh in Figure 4(a) the differences are minor as several nodes are already close to their equilibrium configuration, but for a few triangles the new algorithm still lags behind. With the above settings, both relaxation algorithms do not reach the full equilibrium configuration after a new triangulation in the allotted maximum number of steps N_{long} before the physical timestep is resumed, which is noticeable as a low frequency (corresponding to the time interval between new triangulations) wobble in the mesh.

By increasing the parameters $N_{short}=20$, $N_{long}=60$ and $N_{tri}=20$ for the new relaxation, there are enough steps so that the grid points reach their equilibrium position before the next physical timestep. In Figure 4(d), the triangles for the new algorithm are now much closer to an equilateral shape compared to the old relaxation. Further, Figure 4(b) shows that the new algorithm also

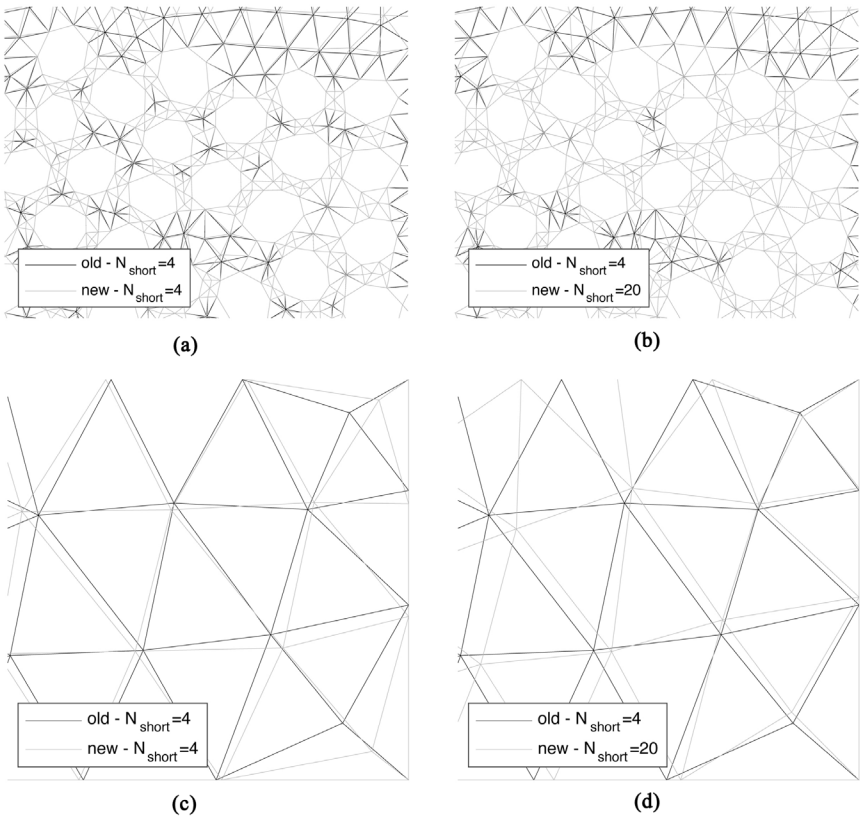


Figure 4. Relaxation of the graded mesh around a granular step as shown in Figure 1. Subfigures (a) and (b) show details of the fine area while (c) and (d) show the coarse area. In (a) and (c), old (black lines) and new (grey lines) relaxation are compared for the same number of relaxation steps. In (b) and (d), the new algorithm uses more relaxation steps and reaches the equilibrium positions faster than the old algorithm.

allows more equilateral configurations in regions of the fine mesh that were not fully equilibrated before. With the nodes now in their equilibrium positions before the physical timestep is resumed, the wobble effect mentioned above is gone. This is in stark contrast to the old algorithm: With a similarly increased step count, oscillations would have time to build up and result in a faulty mesh or squeezed mesh elements. Profiling runs have been performed for the old relaxation with few steps and for the new relaxation with the increased step count. No discernable difference over the CPU-time required for computing a physical timestep was observed, which implies a negligible impact of the new relaxation algorithm, even though it performs more iteration loops. Further, while no change in CPU-time could be observed, the average physical timestep size increased slightly (albeit only about one percent) which warranted further investigation as discussed in the following.

4.2. Effect on the Timestep Control

For simulations of dry granular materials, adaptive time integrators are not very efficient: The timestep is essentially determined by the oscillation frequency due to the particle masses and Young's moduli. In configurations with fluids and particles, the time scale of the dynamics can change much more due to changing motion patterns (release of avalanches, easing of movement after the collapse of aggregates etc.), and even the particle-particle interactions can be "damped away" from the dry oscillation frequency by the viscosity of the fluid in the porespace. In our simulation, we have implemented the time-adaptive BDF2 (backward differentiation formula/Gear Predictor-Corrector of second order) recommended in [7]: As parameter which is used to control the increase or decrease of the physical timestep size, the deviation of the field variables between corrector- and predictor-step is used. Important for the timestep selection is that BDF2 is an implicit integrator, so that the timestep can be selected independently from the grid size and related Von-Neumann stability considerations. Only the maximum timestep limit set by the DEM needs to be considered but, due to the dampening effects of the fluid, the timestep can increase by a factor of about two to five, depending on domain geometry and fluid viscosity. Nevertheless, for years we were rather disappointed that the timestep had been set by the algorithm in a rather "defensive" way: The size of the timestep seemed much slower than the actual dynamics (movement of the particles) mandated and unconnected to the "slowness" of the fluid flow. Only recently, we discovered that among the causes for the small timestep were the spurious grid oscillations mentioned in Section 4.1 caused by the old relaxation in Equation (5). They in turn led to spurious changes of the mesh triangles and therefore the flow rate between some meshes [5]. This introduces considerable noise for the affected triangles, as the flow is interpolated from the old grid and used in the predictor, while the corrector works with the new, modified mesh. As the timestep control must be coupled to the largest overall deviation between predictor- and corrector-step even for even few small meshes, the impact on the timestep is considerable.

As the new relaxation algorithm shows a far more stable behaviour, we prepared a test simulation with nine particles sedimenting in water to compare the effects of new and old relaxation on the timestep (see Figure 5): While the initial configuration is identical for both cases, at the later stage, the new relaxation has farther advanced in physical time, so that the particles have sedimented further. This indicates that the simulation with the new algorithm is running at a larger timestep size. To verify this, a look at the detailed timestep sizes in Figure 6 is necessary: In the first part of the simulation, the timestep sizes start out similar,

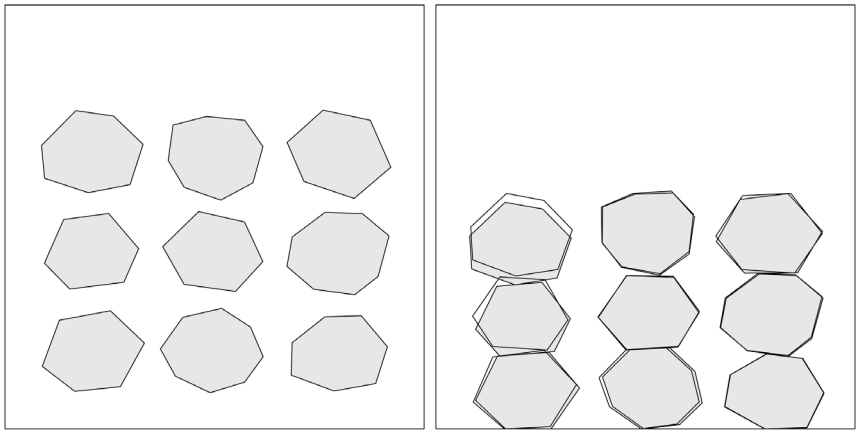


Figure 5. Overlay of nine particles sedimenting in a fluid, simulated with old (only polygonal outline) and new (filled grey polygons) relaxation algorithms. The bottom edge of the frame indicates a solid boundary. The initial configuration (left) is identical for both cases. After 2100 timesteps (right), the new relaxation has advanced further in physical time than the old relaxation due to the—on average—larger timestep size. The particles in the simulation with the new relaxation are mostly stacked into columns with edge-edge contacts, but for the old method, some particles in the left column are still sedimenting.

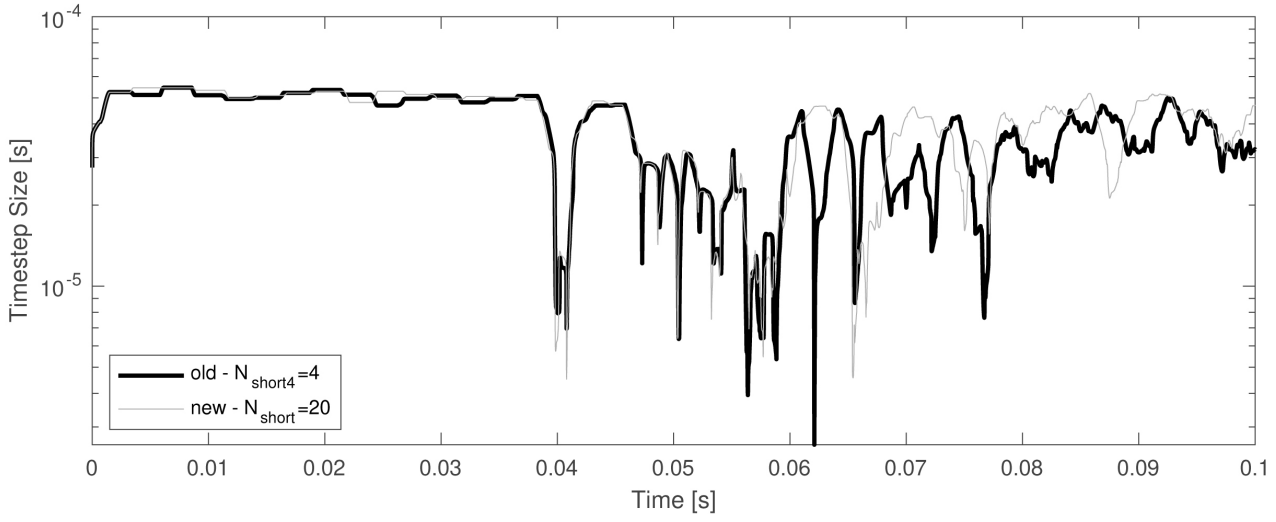


Figure 6. Comparison of physical timestep size for a simulation of nine sedimenting particles with old (thick black line) and new (thin grey line) relaxation algorithms. The displayed timestep size is the moving mean over a 50 timestep interval for better visibility. Note that the timestep size is plotted over physical time and not over the timestep number, so the impact of physical effects (such as particle collisions) can be compared better.

as the timestep remains near the maximum limit defined by the DEM. In this phase, the particles are simply sinking and nothing much of interest is happening. Once the first particle collisions occur, the sudden change in particle velocity causes the length of the timestep to drop. While the timestep for the simulation with the new relaxation algorithm seems to dip a bit further down in case of collisions with high impact forces, it is also able to recover slightly faster and seems to be less affected by collisions with small impact forces. The average timestep size is $\overline{\tau}_{new} = 3.2106 \times 10^{-5}$ for the new relaxation compared to $\overline{\tau}_{old} = 2.9659 \times 10^{-5}$ for the old one, an increase of about 8%. However, the effect on more complex geometries may be less pronounced. If small sections of the pore space prevent a more deformed triangle from relaxing at all (due to geometry constraints), the timestep will be impacted regardless of the relaxation algorithm until the geometry changes again. It should also be noted that in **Figure 6** towards the end, the physical events start to drift apart between the simulations with the different relaxations, even though the timestep values are plotted over physical time. This is caused by the nonlinear-deterministic character of the DEM: Polygonal particles have a high Lyapunov exponent, so e.g. a contact of a corner with an edge allows a particle to tilt either way, left or right. For multiple particles and collisions, this will cascade into an enormous amount of possible final configurations. As a consequence, single particle trajectories have less significance than statistical averages like sinking speeds and slope angles [8]. This gets even more pronounced, when variable timestep sizes come into play. In case of the nine sinking particles, the simulations start to drift apart from each other after a few collisions and from about 0.12 s onwards, the particle positions are distinctly different. The differences in geometry then result in different mesh configurations, meaning a further comparison of timestep size would not yield meaningful data anymore.

To stress-test the relaxation, we modified the above simulation, so the nine particles sink through a greater height difference, resulting in higher particle velocities and thus more intense collisions. The simulation with the old relaxation algorithm failed even before any particles hit the ground: Once the particles reach a certain sinking speed, the relaxation struggles to keep the mesh nodes moving together with the particles until a node in front of a particle gets pushed inside it, causing the simulation to crash, which is the reason why in **Figure 7** the black line suddenly terminates. This can be remedied by increasing the relaxation step counts or by performing new triangulations more often, but this reintroduces mesh oscillations. On the other hand, the new algorithm has no issue moving the mesh together with the sinking particles. For the new relaxation, the grey line in **Figure 7** also shows, the dips in the timestep size caused by the first row of particles bouncing off the floor are only slightly more severe than in **Figure 6**. However, the moment a particle from the second row crashes into a bouncing particle from the first row, the simulation with the new algorithm locks as well. As stated earlier in this section, the upper limit for the timestep is

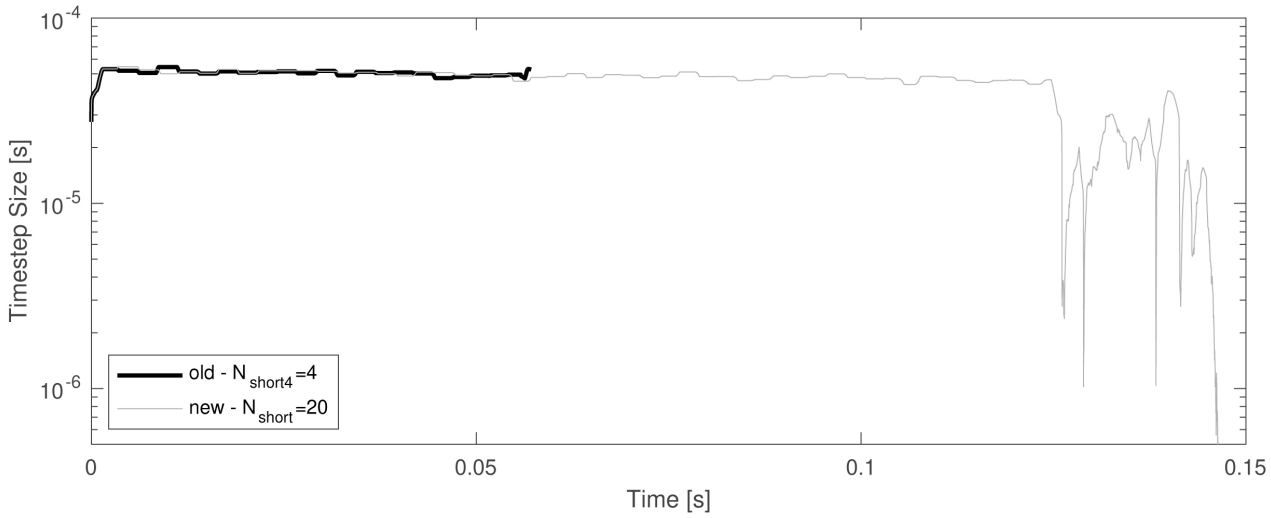


Figure 7. Comparison of physical timestep size for a simulation of nine particles sedimenting from a greater height. The old relaxation (thick black line) causes the simulation to crash due to a meshing error. The simulation with the new relaxation algorithm (thin grey line) locks because of a residual error, once the DEM starts feeding unphysical jumps in the particle velocities into the FEM. The displayed timestep size is again the moving mean over a 50 timestep bracket, and plotted over the physical time.

given by the DEM but increased by a factor as the particle movement is damped by the fluid. With the increased particle velocities and even higher relative velocities due to the bouncing, the dampening effect of the fluid was insufficient. With the timestep at the moment of the collision still well above the limit for a dry DEM, the calculation of the contact forces resulted in unphysical values for the particle velocities which, when fed back into the FEM, locked the simulation. This of course can be remedied by reducing the maximum possible timestep size, but it is a great example that the new relaxation allows us to keep the timestep size limit coming from the FEM consistently above the one from the DEM, meaning timestep size will for the most part not be limited by the difference between predictor and corrector results anymore.

5. Summary and Conclusions

We used a spring-model between grid points to improve the quality of the FEM mesh generated via Delaunay triangulation: The relaxed length of a spring is matched to the side length of equilateral triangles, equilibrating all mesh triangles towards a more equilateral shape. An existing zero-order integrator was used to compute the equilibration process and by improving its convergence properties, we were able to suppress oscillations in the equilibration more efficiently and allow for the equilibrium state to be reached faster. The details of the grid relaxation do not influence the FEM-solution directly; they just increase the speed at which the quality of the mesh is improved. Nevertheless, as a "bad mesh" in the pore spaces between granular particles leads to ill-conditioning of the underlying nonlinear system of equations, the improved mesh generation helps to increase the overall stabilisation of the simulation and reduces the risk of faulty mesh structures locking the simulation.

For some setups, the possible physical timestep was increased by up to 8%, while for particles moving inside the fluid, the relaxation could now keep up with the changes of the particle outlines where the older algorithm failed, while both with respect to memory requirements and computational effort, the additional costs of the new algorithm are negligible.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Matuttis, H.-G. and Chen, J. (2014) Understanding the Discrete Element Method. John Wiley and Sons, Hoboken. https://doi.org/10.1002/9781118567210

[2] Mueller, J., Kyotani, A. and Matuttis, H.-G. (2020) Towards a Micromechanical Understanding of Landslides—Aiming at a Combination of Finite and Discrete Elements with Minimal Number of Degrees of Freedom. *Journal of Applied Mathematics and Physics*, **8**, 1779-1798. https://doi.org/10.4236/jamp.2020.89134

[3] Ng, S.H. and Matuttis, H.-G. (2011) Adaptive Mesh Generation for Two-Dimensional Simulation of Polygonal Particles in Fluid. *Theoretical and Applied Mechanics Japan*, **59**, 323-333.

[4] Persson, P.O. and Strang, G. (2004) A Simple Mesh Generator in MATLAB. *SIAM Review*, **46**, 329-345. https://doi.org/10.1137/S0036144503429121

[5] Mueller, J., Kyotani, A. and Matuttis, H.-G. (2021) Grid-Algorithm Improvements for Dense Suspensions of Discrete Element Particles in Finite Element Fluid Simulations. *EPJ Web of Conferences*, **249**, Article No. 09006. https://doi.org/10.1051/epjconf/202124909006

[6] van Kan, J., Vermolen, F. and Segal, A. (2005) Numerical Methods in Scientific Computing. VSSD, Delft.

[7] Gresho, P.M. and Sani, R.L. (2000) Incompressible Flow and the Finite Element Method, Volume 2: Isothermal Laminar Flow. John Wiley and Sons, Hoboken.

[8] Krengel, D. (2019) Numerically Exact Computation of Static Friction for Many-Body Problems. Ph.D. Thesis, The University of Electro-Communications, Chofu.