ITM Web of Conferences **2**, 03001 (2014)
DOI: 10.1051/itmconf/20140203001

Dynamic mesh optimization based on the spring analogy

Jonas Schmidt[a] and Bernhard Stoevesandt

Fraunhofer Institute for Wind Energy and Energy System Technology IWES, Oldenburg, Germany

Abstract. We present an implementation of the spring analogy for three dimensional meshes in OpenFOAM. All parameters of the spring system are treated as fields that can either be pre-defined by the user, or updated at each time step according to specified geometrical regions or diffusion equations. The purpose of the method is to provide a pre-processing tool for mesh optimization. We study three simple test cases, a deformed block, an airfoil and a hill, and we analyze the evolution of skewness, non-orthogonality and aspect ratio during the approach of dynamic equilibrium.

1. Introduction

Mesh quality is essential for computational fluid dynamics (CFD) simulations, and yet remains a time consuming if not frustrating issue in many cases. Complex geometries often require unstructured meshes, and the corresponding meshers are not always tailored to work easily for the cases of interest. Here we present a pre-processing tool for OpenFOAM [1] (version 2.1.1) that aims at improving the quality of an existing mesh based on a physical principle.

This principle is dynamical equilibrium of a system of point particles, subject to nearest-neighbour forces. In the easiest approach followed here, these forces are springs of vanishing or finite equilibrium length and a given stiffness. The spring system is obtained by interpreting the mesh points as dynamical point objects of unit mass, and the edges as springs interconnecting them.

For unstructured triangular meshes around airfoils this idea was first studied by Batina [2], and revisited by many authors ever since (for a review see for example [3]). Often the main interest lies in the moving boundary problem, where during run-time of the simulation the geometry or its relative position with respect to the flow mesh changes. In OpenFOAM, this has been solved effectively based on the principle of diffusion [4–6] by a Laplacian approach (cf. [7]). Here this is not our focus, we are interested in mesh optimization during pre-processing instead.

The spring analogy approach has well known issues with the preservation of the cell topology. Depending on the choices of the parameters and the initial configuration, straight-forward implementations may show edge collapse and face or cell inversion. Batina [2] proposed linear springs with a stiffness inversely proportional to the spring length to prevent node collision. A successful generalization of the idea to modify the spring constant in order to achieve stability are torsional springs, developed by Nakahashi and Deiwert [8]. Farhat *et al.* used and generalised this approach

[a]e-mail: jonas.schmidt@iwes.fraunhofer.de

for triangulated meshes in two [9] and three dimensions [10]. Another approach within the framework of a linear spring system is the ball-vertex method [11], where additional springs are added that resist the motion of a vertex to its opposite face. For more work on the stabilization of spring systems and other self-adaptive mesh motion methods, we refer the reader to [3, 7, 12] and references therein.

Here a different approach is followed, based on parameter fields that allow for case tailored dynamic mesh solutions. The benefit of the approach is that specific requirements on the mesh, like local refinement in selected regions, can be obtained by adjusting the parameters that enter the dynamics. The drawback is that the stability of the method depends on the parameter choices, and thus the approach can be expected to be less robust in general.

In Section 2 the physics of the underlying spring system is revisited, and a brief summary of its implementation in OpenFOAM is given in Section 3. Section 4 then presents three test case examples for the method, a deformed block, an airfoil and a smooth hill. We finally discuss the results and conclude in Section 5.

2. The spring analogy

Meshes for CFD calculations consists of a finite number of cells which are organized in a specific topology. Each cell can be defined by a set of faces that fully encloses a finite and non-zero volume. Each face is defined by a finite number of points, and each of the points is connected to two other points of the same face by an edge.

The spring analogy is based on the idea of mapping a mesh to a system of point objects connected by springs. This is achieved by identifying the mesh points with the dynamic point objects and the edges of the mesh with the springs, whose properties are to be specified. The dynamical system is then solved for its equilibrium configuration. Ideally, the latter can be back translated to a mesh with improved cell quality.

The defining property of a spring is the fact that the restoring force $\mathbf{F}$ is proportional to the displacement from the equilibrium position. For the spring between objects A and B with positions $\mathbf{x}_\mathrm{A}$ and $\mathbf{x}_\mathrm{B}$, the force acting on object A is

$$\mathbf{F}_\mathrm{AB} = k\,(|\mathbf{x}_\mathrm{B} - \mathbf{x}_\mathrm{A}| - l_0)\,\mathbf{n}_\mathrm{AB}, \qquad \mathbf{n}_\mathrm{AB} = \frac{\mathbf{x}_\mathrm{B} - \mathbf{x}_\mathrm{A}}{|\mathbf{x}_\mathrm{B} - \mathbf{x}_\mathrm{A}|}\,. \tag{1}$$

Here l_0 is the equilibrium length of the spring and k the spring stiffness. The force induced by object A onto object B through the spring is $\mathbf{F}_\mathrm{BA} = -\mathbf{F}_\mathrm{AB}$. The total force acting on object A due to edge springs is then

$$\mathbf{F}_\mathrm{A}^\mathrm{edges} = \sum_{\text{B edge}-\text{connected to A}} \mathbf{F}_\mathrm{AB}. \tag{2}$$

The discrete dynamics of object A, subject to the force $\mathbf{F}_\mathrm{A}$, can be expressed in terms of the velocity $\mathbf{v}_\mathrm{A}$,

$$\mathbf{v}_\mathrm{A}^{n+1} = (1 - \mu_\mathrm{A})\,\mathbf{v}_\mathrm{A}^{n} + \Delta t\,\frac{\mathbf{F}_\mathrm{A}}{m_\mathrm{A}}\,. \tag{3}$$

Here μ_A is a dimensionless constant bound by 0 and 1, modelling friction, n labels the time step, and Δt is the respective time increment. For a review on mesh spring systems and their numerical solution, see eg. [3]. In the approach discussed here, the parameters k, l_0, μ are treated as independent fields, and Δt is a constant. The mass field m can be absorbed into the stiffness fields, and is thus set to unity in the following.

For each spring a core with width l_core is defined. The spring force from Eq. (1) is replaced by

$$\mathbf{F}_\mathrm{AB} = \begin{cases} k\,(l - l_0)\,\mathbf{n}_\mathrm{AB}\,, & \text{if} \quad l \geq l_\mathrm{core}, \\ \left[k\,(l - l_0) + k_\mathrm{core}(l - l_\mathrm{core})\frac{l_\mathrm{core}}{l}\right]\mathbf{n}_\mathrm{AB}, & \text{if} \quad l < l_\mathrm{core} \end{cases} \tag{4}$$

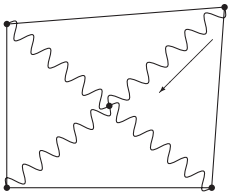

Figure 1. Springs connecting the centre of a mesh element with its vertices. Their equilibrium length is set to the mean distance of the vertices from the centre.

where $l = |\mathbf{x}_B - \mathbf{x}_A|$. This force diverges as l approaches zero. Note that for $0 < l < l_{core} < l_0$ the factor inside the square brackets is negative and thus object A experiences a repelling force. Thus edge collisions can be prevented, with a repulsive force of variable strength.

Additional to the edge springs, further stabilizing forces from face and cell centres to the corresponding vertices are introduced, as sketched in Figure 1. They follow the spring law (1), with equilibrium length l_0 given by the mean of the distances from the mesh element centre to the vertices. The corresponding spring constants are denoted as k_{face} and k_{cell}, respectively. Core regions are defined similar to Eq. (4),

$$\mathbf{F}_{el,\,core} = k_{el,\,core}(V_{el,\,core} - V_{el})\frac{r_{el,\,core}}{V_{el}}\mathbf{n} \qquad \text{added if } V_{el} < V_{el,\,core}, \tag{5}$$

where the subscript 'el' represents 'face' or 'cell', $\mathbf{n}$ points from the centre to the vertex, and V_{el} is the volume of the element. The core volume is in both cases determined by a radius $r_{el,core}$,

$$V_{face,\,core} = \pi\, r^2_{face,\,core}, \quad V_{cell,\,core} = \frac{4}{3}\pi\, r^3_{cell,\,core}\,. \tag{6}$$

In total, eight additional parameter fields were introduced, k_{core}, l_{core}, k_{face}, $k_{face,\,core}$, $r_{face,\,core}$, k_{cell}, $k_{cell,\,core}$, $r_{cell,\,core}$. They are available for additional control of the mesh dynamics, if necessary.

3. Implementation

3.1 Dynamic mesh solver

A new dynamic mesh solver was implemented for OpenFOAM 2.1.1, solving Eq. (3) for each time step. The name of the solver is `pointForceMotionSolverDynamics`, it is derived from OpenFOAM's `fvMotionSolver`. It reads the mass, friction and total point force fields, all stored with support on the mesh points.

The point force field has three underlying sources: The edge-supported spring force field and the face and cell stabilization force fields. All three yield contributions to the total force at each node and are updated at each time step.

3.2 Parameter field manipulation

The stiffness and the equilibrium spring length fields are spring specific data, and therefore stored with support on the edges of the mesh. For convenience they are combined from an isotropic and an axial contribution, both stored as point-based scalar data fields. In addition, a vector field $\mathbf{n}_{axial}$ with point support is defined. The axial contribution to the spring properties is then calculated by projection of the edge vector onto $\mathbf{n}_{axial}$, and multiplication with the mean value of the axial scalar field at the corresponding edge points.

Three methods are implemented for manipulating the parameter fields, including isotropic and axial edge property data and the axial vector field $\mathbf{n}_{\text{axial}}$:

1. The fields can be read from files and then be treated as constant.
2. The fields can be recalculated at each time step, given specific values in pre-defined geometrical regions. The list of cells that are inside the fixed regions is updated based on an algorithm that checks all neighbour cells, as long as the cell centre is found to lie inside.
3. The fields can be solutions of diffusion equations, updated at each time step.

The latter method may for example be applied to the axial direction field $\mathbf{n}_{\text{axial}}$, which can thus be set up to follow normals from one patch to its opposite counterpart.

3.3 Boundary conditions

The following OpenFOAM boundary conditions for the vector field that defines the mesh point motion are supported:

- **`fixedValue`**: Usually the displacement vector is set to zero on the boundary.
- **`calculated`**: The boundary points are let free, the boundary thus deforms subject to the spring forces acting on the patch points[1].
- **`fixedNormalSlip`**: Given the constant and homogeneous normal vector of the patch, the corresponding component of the displacement is ignored. This only works well for planes.

For non-plane boundary geometries, a new boundary condition was implemented:

- **`tangentialSlide`**: The initial geometry is stored in memory, and each patch point is associated with a nearest face on that original geometry. After each infinitesimal movement it is projected back onto the geometry, by a divide and conquer algorithm with respect to that nearest face. The latter is updated based on a neighbour search at each time step. An optional parameter can be specified to realise a smooth transition between the `calculated` and the `tangentialSlide` boundary conditions.

4. Examples

4.1 A deformed block

A first test for the spring analogy is the deformed block from Figure 2. The initial mesh has $20 \times 12 \times 10 = 2400$ cells, maximal extension of $(1000, 500, 300)$ m in x, y, z direction, respectively, and simple grading factors $(15, 1, 1)$.

All boundary conditions were chosen to be `calculated`, so the mesh is freely moving under the influence of the spring forces. All parameter fields were uniform, with the following values:

- Equilibrium spring length: $l_0 = 30\,\text{m}$,
- Spring stiffness: $k = k_{\text{face}} = k_{\text{cell}} = 1\,\text{N}\,\text{m}^{-1}$,
- Friction: $\mu = 0.08$,

All spring core coefficients were set to zero. The time increment was $\Delta t = 0.1\,\text{s}$, the result shown in Figure 2 was obtained after 700 time steps. Clearly the springs have driven the mesh towards its equilibrium configuration. All deformation and grading features have disappeared in the converged solution.

[1] Free dynamics may result in global mesh movement, if the spring forces are not balanced. This can be prevented by locally enforcing infinite mass for selected mesh points.

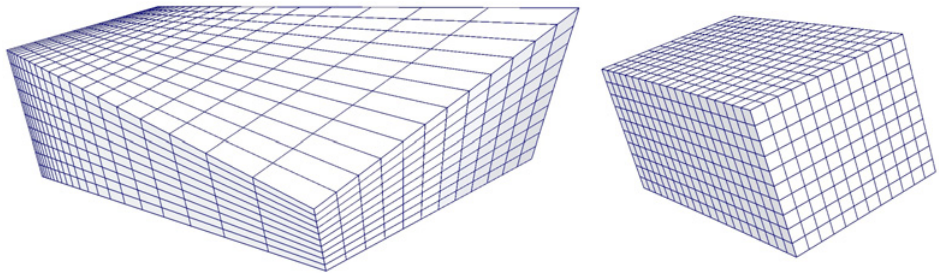

Figure 2. The `calculated` boundary condition for the example of a deformed block. Left: the initial mesh. Right: the final mesh.

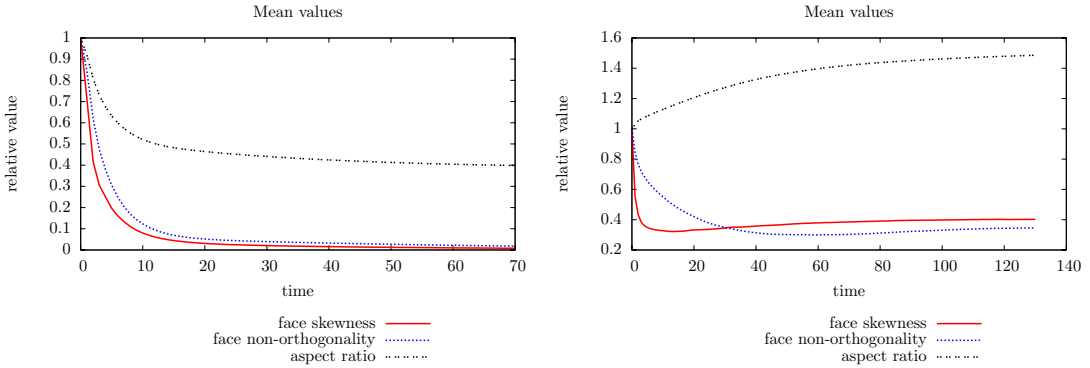

Figure 3. The normalized evolution of skewness, non-orthogonality and aspect ratio. Left: The deformed block example from Figure 2. Right: the complex terrain example from Figure 6.

This impression is in agreement with the quantitative results shown in the left panel of Figure 3. One clearly observes a reduction of the mesh quality measures skewness, non-orthogonality and aspect ratio for the initially deformed block. Also note the fast convergence for the case of this simple and purely academic example.

4.2 A wind turbine airfoil

A C-type mesh around an airfoil was used for demonstrating the effect of inhomogeneous cell-isotropic spring stiffness, cf. Figure 4. The mesh has 2030 cells in each of the 4 layers perpendicular to the height direction z. The total mesh extension is $11 \times 10 \times 1$ chord lengths in (x, y, z) directions, and the chord length is 1 m. The trailing edge was cut at the end. The following parameters were chosen for the mesh dynamics:

- Equilibrium spring length: $l_0 = 0.03\,\mathrm{m}$,
- Spring stiffness:
  - $k = 10\,\mathrm{N\,m^{-1}}$ in a cylinder of radius 1 m centred 0.5 m from the trailing edge,
  - $k = 25\,\mathrm{N\,m^{-1}}$ in a cube of side length 1 m including the trailing edge region,
  - $k = 20\,\mathrm{N\,m^{-1}}$ in a cuboid of width and depth 1 m along the straight line from trailing end to the outflow region,
  - $k = 3\,\mathrm{N\,m^{-1}}$ elsewhere,
- Friction: $\mu = 0.2$.

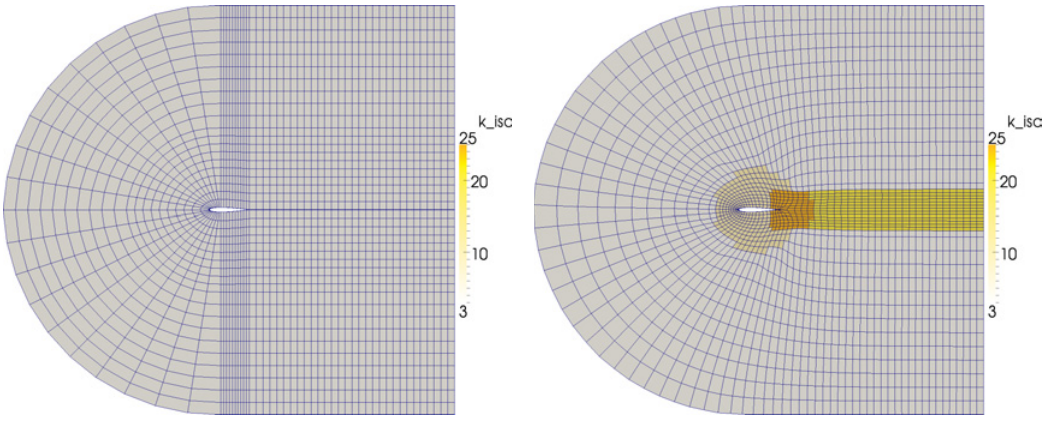

Figure 4. The `tangentialSlide` boundary condition combined with inhomogeneous isotropic spring stiffness, for the example of an airfoil. Left: the initial mesh. Right: the final mesh.

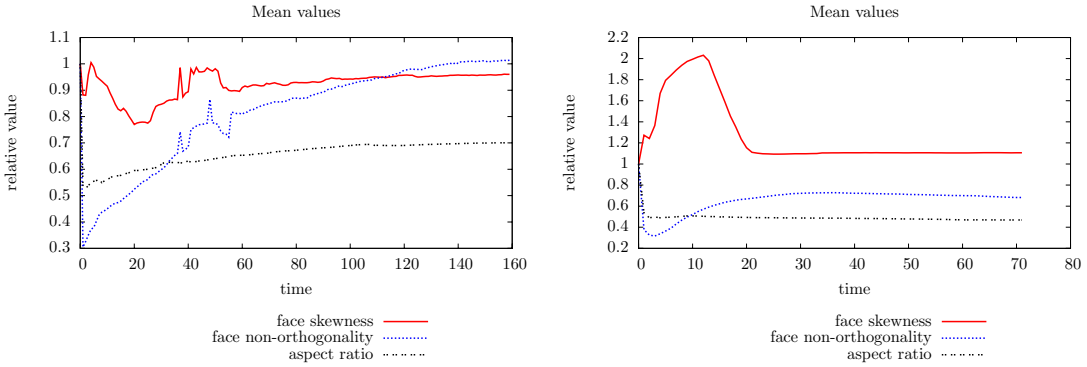

Figure 5. The normalized evolution of skewness, non-orthogonality and aspect ratio for the airfoil example. Left: the case from Figure 4 with inhomogeneous stiffness. Right: the same initial mesh with homogeneous stiffness.

All spring core coefficients and the cell and face stabilizing forces were set to zero. The time increment was $\Delta t = 0.1\,\mathrm{s}$, the result shown in Figure 4 was obtained after 160 time steps. It shows the establishment of smooth boundary layer and wake refinement regions in dynamical equilibrium. Note that both the inflow boundary and the airfoil patch were equipped with the `tangentialSlide` boundary condition.

Figure 5 shows the mesh quality parameters skewness, non-orthogonality and aspect ratio for airfoil case from Figure 4 (left panel). The corresponding mesh parameters for the case of uniform stiffness[2] are shown in the right panel for comparison. The realization of refinement by an inhomogeneous stiffness field is accompanied by reduced mesh quality improvement.

4.3 Complex terrain

The effect of axial stiffness, ie., a non-isotropic spring constant field, is shown in Figure 6 for the case of a structured mesh over a hill. For each time step a Laplace equation was solved to find vector field

[2] For this simulation the smooth interpolation between the boundary conditions `calculated` and `tangentialSlide` was used on the airfoil patch for $t < 32$, and `fixedValue` afterwards.

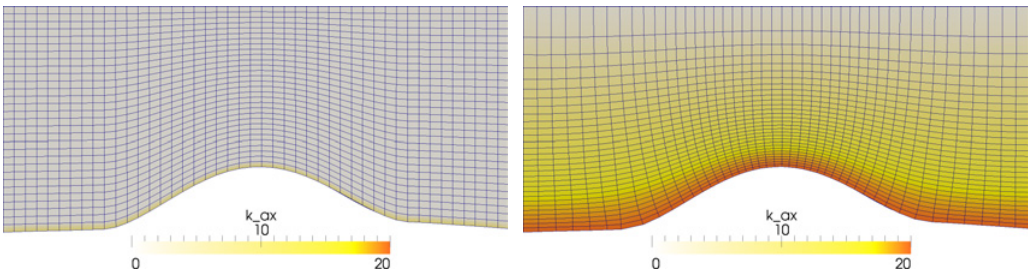

Figure 6. The `tangentialSlide` boundary condition combined with inhomogeneous axial spring stiffness following patch normals, for the example of a hill. Left: The initial mesh. Right: The final mesh.

that carries the normals of the ground patch to the inverted normals of the top patch. Another Laplace equation was solved for a scalar field, with fixed value boundary conditions on these two patches, with uniform values

- ground : $k_{\mathrm{ax}} = 25\,\mathrm{N\,m^{-1}}$,
- top : $k_{\mathrm{ax}} = 0\,\mathrm{N\,m^{-1}}$,

and zero gradient conditions on the sides. For each edge the total spring stiffness k was then set to the projection of the edge direction onto the patch normal transport field, multiplied with the local value of the axial stiffness field k_{ax}.

The test mesh has $4 \times 45 \times 30 = 5400$ cells and total extension of $100\,\mathrm{m} \times 1000\,\mathrm{m} \times 452$ m. The initial structured mesh was created using the open source tool `terrainBlockMesher` [13], which in principle does not rely on spring dynamics for the creation of a boundary layer over complex terrain.

The following values were chosen for the remaining parameters:

- Equilibrium spring length: $l_0 = 0\,\mathrm{m}$,
- Face and cell stabilization: $k_{\mathrm{face}} = k_{\mathrm{cell}} = 1\,\mathrm{N\,m^{-1}}$,
- Friction: $\mu = 0.4$,

All spring cores were set to zero, the time increment was $\Delta t = 0.1\,\mathrm{s}$, and Figure 6 was obtained after 130 time steps. The inhomogeneous spring stiffness field yielded grading in height direction, following the terrain. Furthermore, the equilibrium solution shows splines that are orthogonal to the ground surface. Note that the boundary conditions of both the ground and the top patch were chosen to be `tangentialSlide` in this example.

The mesh quality parameters for this case and their convergence are shown in Figure 3 (right panel). Face skewness and non-orthogonality were efficiently reduced by the spring method. Due to the boundary layer requirement the mean aspect ratio was increased compared to the initial mesh.

5. Discussion

We implemented a new mesh motion solver in OpenFOAM, based on the spring analogy. It is intended as a pre-processing tool for mesh optimization, and the purpose of this paper was to show that the basic principle works as expected on small test cases. Indeed we showed that by using inhomogeneous spring stiffness fields, the method is capable of creating surface boundary layers and mesh refinement regions as desired.

Our guide during this work was the idea that the dynamical equilibrium of a physical system may have beneficial effects on the quality of a mesh, whose purpose it is to provide the basis for numerical

flow simulations with good convergence properties and high quality results. We are aware of the fact that this is a hypothesis, and issues one may have with CFD meshes do not necessarily have a counterpart that can be addressed by dynamic equilibrium.

The strength of the method is that skewness, orthogonality and aspect ratio issues can in principle be cured or alleviated by the spring system approach. The general tendency of the spring dynamics is to smoothen the properties of the mesh on global and local scales. This, however, also yields the possibility that refinement regions are washed away during the approach of dynamical equilibrium in homogeneous conditions. This is physically expected for the spring system, but not desired for CFD meshes. We address this problem by treating all spring and point properties as possibly inhomogeneous fields.

Our pre-processing tool is therefore based on various parameter fields that can be specified prior to the spring system simulation. Alternatively they can be updated at each time step, according to the varying cell content of pre-defined fixed geometrical regions, or solutions of diffusion equations. Especially the option to define inhomogeneous spring stiffness appears to be efficient for the creation of boundary layers and refinement regions.

We tested the approach of inhomogeneous spring stiffness fields in two examples, a C-type mesh around an airfoil and a structured mesh over a smooth hill. In both cases we observed that the method was capable of creating the desired refinement regions. However, these regions have an influence on the mesh quality parameters, since they might enforce stronger deformation conditions on the mesh. In the complex terrain example, the increased aspect ratio at the top of the domain is a direct consequence of the desired boundary layer near the ground patch, compared to the homogeneous stiffness solution. In the airfoil case, the transition between two different stiffness regions may have caused skewness and non-orthogonality. This might be improved by imposing smoothing routines on the inhomogeneous stiffness field, leading to more continuous properties of the dynamic spring system.

Our test examples showed that the spring equilibrium length is a parameter that should be treated with care. In general the spring system behaves unstable if the equilibrium configuration relies on compressive stress. This is intuitive, since small deviations will immediately imply stress release, followed by cell destruction. In contrast, with spring systems under tensile stress we observe rather robust behaviour and good dynamical convergence.

In summary the tool is susceptible to instabilities due to the freedom to set up the parameter fields, but it is precisely this option which allows one to incorporate experience and the understanding of the underlying physics and numerics into the mesh optimization. Future work will be concerned with the parallelization of the method and its application to realistic cases, including a comparative CFD mesh study.

References

[1] OpenFOAM, `http://www.openfoam.org` (2013), [Online; accessed 18-February-2013], `http://www.openfoam.org`

[2] J. Batina, AIAA Journal **28**, 1381 (1990)

[3] F. Blom, Int. J. Numer. Meth. Fluids **32**, 647 (2000)

[4] H. Jasak, Z. Tuković, *Automatic mesh motion for the unstructured finite volume method* (2004)

[5] T. Lucchini, G. D'Errico, H. Jasak, Z. Tukovic, SAE Technical Paper 2007-01-0170 (2007)

[6] Z. Tukovic, H. Jasak, Computers and Fluids **55**, 70 (2012)

[7] C. Burg, International Journal for Numerical Methods in Fluids **52**, 163 (2006)

[8] K. Nakahashi, G.S. Deiwert, AIAA Journal **25**, 513 (1987)

[9] C. Farhat, C. Degand, B. Koobus, M. Lesoinne, Comput. Meth. Appl. Mech. Eng. **163**, 231 (1998)

[10] C. Degand, C. Farhat, Computers and Structures **80**, 305–316 (2002)
[11] C.L. Bottasso, D. Detomi, R. Serra, Computer Methods in Applied Mechanics and Engineering **194**, 4244 (2005)
[12] G. Markou, Z. Mouroutis, D. Charmpis, M. Papadrakakis, Computer Methods in Applied Mechanics and Engineering **196**, 747 (2007)
[13] J. Schmidt, `https://github.com/jonasIWES/terrainBlockMesher.git` (2013)