

# A Review of Shape Distortion Methods Available in the OpenFOAM<sup>®</sup> Framework for Automated Design Optimisation



Steven Daniels, Alma Rahat, Gavin Tabor, Jonathan Fieldsend and Richard Everson

**Abstract** Parametrisation of the geometry is one of the essential requirements in shape optimisation, and is a challenging subject when carrying out a automated procedure. It is critically important to maintain the consistency of the shape and grid quality between each evaluation, while providing flexibility for a wide range of shapes using the same parameterisation of the geometry. The sensitivity of the grid to the changes to the geometry must be at a minimum during this process. This contribution presents a review of the grid distortion and regeneration methods available within the OpenFOAM<sup>®</sup> framework which can be utilised for shape optimisation. The objective of this contribution is to compare the effectiveness of these methods in the automated procedure and to provide suggestions for improvements. Special attention is given to three major factors involving shape optimisation: automation of model abstraction, automation of grid deformation or regeneration and robustness.

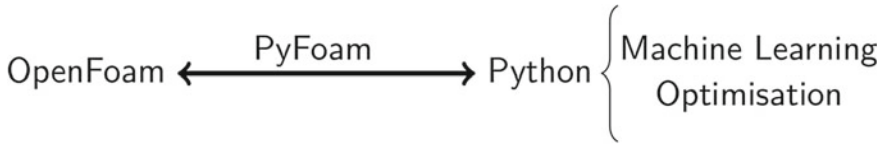
## 1 Introduction

Optimisation of designs using Computational Fluid Dynamics (CFD) is frequently performed across many fields of research, such as the optimisation of an aircraft wing to reduce drag or an increase in the efficiency of a heat exchanger. General optimisation strategies involve modification of design variables with a view to improving the appropriate objective function(s). Often, the objective function(s) is (are) nonlinear and multi-modal, and hence polynomial time algorithms for solving such problems may not be available. In such cases, applying Machine Learning methods such as Evolutionary Algorithms (EAs—a class of stochastic global optimisation technique inspired by natural evolution) may bring to light good solutions within a practical time frame. The traditional CFD design optimisation process is often based on a ‘trial-and-error’ type of approach. Starting from an initial geometry, Computationally Aided Design changes are introduced manually based on results from a limited number of design evaluations and CFD analyses. The process is usually complex,

---

S. Daniels (✉) · A. Rahat · G. Tabor · J. Fieldsend · R. Everson  
University of Exeter, Harrison Building, North Park Road, Exeter EX4 4QF, UK  
e-mail: [S.Daniels@exeter.ac.uk](mailto:S.Daniels@exeter.ac.uk)

© Springer Nature Switzerland AG 2019  
J. M. Nóbrega and H. Jasak (eds.), *OpenFOAM*<sup>®</sup>,  
[https://doi.org/10.1007/978-3-319-60846-4\\_28](https://doi.org/10.1007/978-3-319-60846-4_28)



**Fig. 1** Framework developed in the present work for combining OpenFOAM<sup>®</sup> with various Python-based Machine Learning libraries [1]

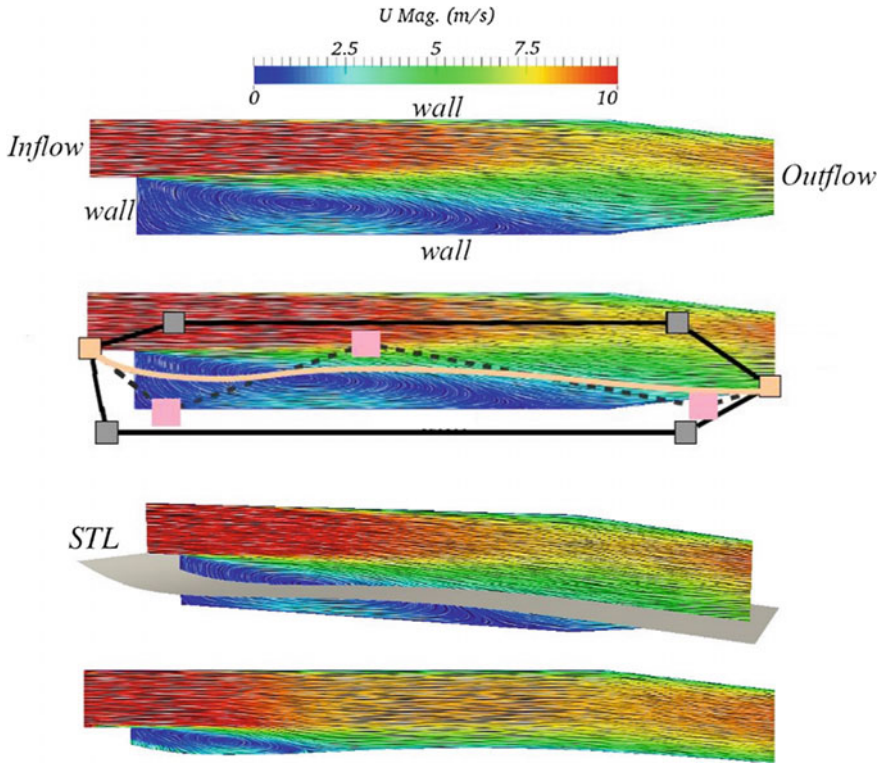
time-consuming and relies heavily on engineering experience, hence the overall design procedure is often inconsistent, i.e. different ‘best’ solutions are obtained from different designers.

Based on the limitations of optimisation by hand, using EAs and CFD simulations may be an attractive alternative. There have been other attempts to combine EAs and CFD simulation to optimise design (see for example, [2]). Although EAs do not guarantee the determination of the optimal solution, they may achieve good solutions consistently. In this present work, an automated framework combining Python’s EA library DEAP with OpenFOAM<sup>®</sup> 2.3.1 was developed. The communication of the Python libraries with OpenFOAM<sup>®</sup> was achieved using PyFoam 0.6.5. PyFoam was used as an interface to control the OpenFOAM<sup>®</sup> case set-ups for each proposed solution from the EA code and to post-process the data generated after each CFD calculation. A summary of this framework can be seen in Fig. 1.

While the aim of this project is to focus on complex (potentially multi-objective) cases, for the purpose of this contribution, a single-objective optimisation of the PitzDaily tutorial in OpenFOAM<sup>®</sup> was used to demonstrate the performance of the proposed procedure. The chosen cost function for this case was to minimise the static pressure drop between the inflow and outflow boundary conditions, i.e.

$$f_1 = \min(|p_i - p_o|), \quad (1)$$

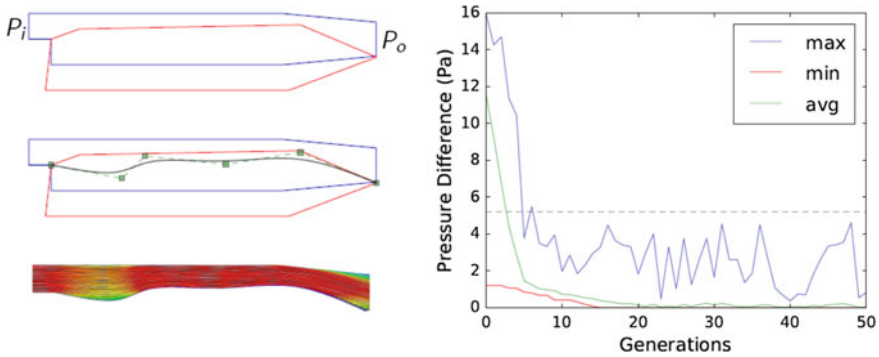
where both pressures were obtained by averaging over the boundary condition faces. Figure 2 (top-to-bottom) shows the procedure in which the shape is altered after each evaluation. To change the geometry, subdivision curves [3] were generated to form a new lower wall. The grey squares indicate the coordinates for the search space (or bounding box), with the black lines showing the resulting boundary. The fixed points of the spline (indicated by orange squares) were placed across the bottom boundary condition (‘lowerWall’). The squares in pink indicate the control points that may be altered by the EA, and thus change the curvature of the spline. After the new positions of the pink squares were proposed by the EA, a Stereolithography (.stl) file was generated and passed to OpenFOAM<sup>®</sup>. Using this file, the meshing utility snappyHexMesh was used to cut the domain away (and, if required, re-mesh the altered region). Subsequently, the case was run using the steady-state solver simpleFoam. After this, the cost function was obtained and the EA determined a new position for the coordinates of the pink squares.



**Fig. 2** Shape optimisation procedure of the PitzDaily tutorial case

A single-objective optimisation run of the PitzDaily tutorial was performed with a basic real-valued Genetic Algorithm [4]. For the CFD calculation, the residual tolerances for the SIMPLE algorithm for velocity and pressure were set to  $10^{-5}$  and  $10^{-6}$ , respectively. The pressure drop calculated from the original (base) case was measured as  $|\Delta p| = 5.22$  Pa. Figure 3 (left) shows the optimised shape of the PitzDaily geometry. The resulting pressure drop was calculated as  $|\Delta p| = 4.7 \times 10^{-6}$  Pa. The schematic diagrams above the contour diagram of velocity magnitude indicate the position of the control points, and the outer boundary region (in red) that they must not exceed. The GA maintains a population of solutions: Fig. 3 shows the maximum (worst), the average and the minimum (best) pressure drop across the population with each generation. The grey-dashed line indicates the pressure drops for the original PitzDaily geometry. It can be seen that the reduction of the best pressure difference between the inflow and outflow was reached after 15 generations, and an improvement of the pressure difference from the original case was achieved sooner than this.

For shape optimisation problems, the design variables usually involve parameters that modify the shape of a given initial geometry. For a certain objective function,



**Fig. 3** Single-objective optimisation of the PitzDaily tutorial case in reducing the pressure drop

the design variables are used to define a deformed geometry at each optimisation cycle, which is used to compute the new flow field that is required to evaluate the objective function. In the traditional body-fitted approach, on structured or unstructured meshes, a change in the shape of the surface mesh requires a smooth transition in its deformation and the avoidance of large distortions and interpenetration of neighbouring elements in the CFD mesh. Various mesh deformation methods (e.g. spring stiffness, elastic analogy [5]) have been proposed to tackle this problem, but as the geometric complexity of optimisation problems increases, the robustness of this approach reduces; the resulting computational cost of the mesh deformation is not negligible. In addition, the linearisation of the mesh distortion scheme (i.e. mesh sensitivity) must be computed to take into account the effect of shape perturbations on the flow equations, with additional cost and complexity.

In general, mesh generation is commonly recognised as one of the main challenges in CFD. Mesh quality issues can significantly impact the accuracy of the eventual solution, even to the point at which the solver diverges and no solution is generated; they can also significantly affect the level of computational work (e.g. number of evaluations) necessary to reach the solution. Modern Finite Volume (FV) CFD codes tend to use arbitrary unstructured or polyhedral meshes, allowing for a wide variety of cell shapes to accommodate complex geometries. This also allows for a wide variety of mesh problems; non-orthogonality, face skewness etc., and whilst modern solution algorithms can typically correct for mild levels of mesh problems, this is at the cost of increased numerical error. Pathological levels of mesh problems can lead to algorithm divergence. The acceptable level of mesh quality also varies according to the details of the modelling being used; for example, the turbulence modelling in Large Eddy Simulation (LES) ties in very closely with aspects of the mesh, such as cell size, thus requiring much higher levels of mesh quality than for cases involving Reynolds-Averaged Navier–Stokes (RANS) [6]. Note that, our discussion revolves around issues relating to mesh generation for FV CFD, which is in our area of familiarity. Similar issues undoubtedly arise for Finite Element (FE) methods and

other applications of these techniques. The remainder of this contribution will review the basic methods available in OpenFOAM®, and their effect on the reliability of the solution with respect to the quality of the mesh.

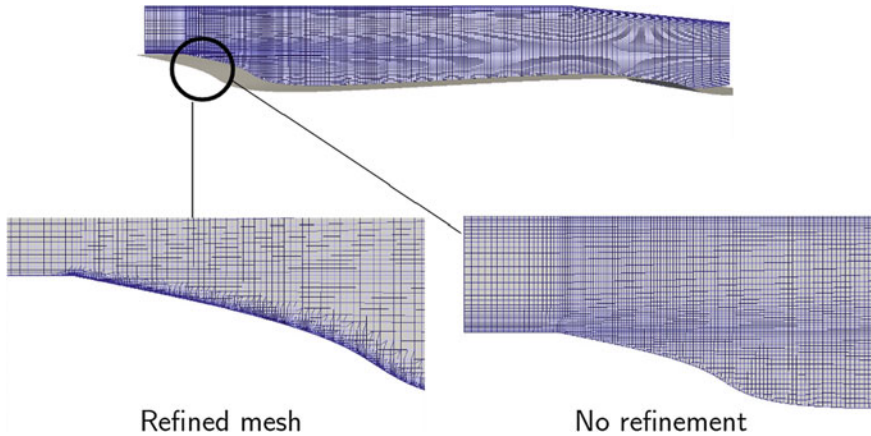
## 2 Grid Deformation and Regeneration Techniques

During the design optimisation process, the design surfaces are perturbed after each evaluation. These perturbations must be transferred from the surface points to the grid in the surrounding flow field. To achieve this, the methods available can be placed into two categories: grid regeneration and grid deformation; the latter has previously been demonstrated in OpenFOAM® by [7]. The techniques reviewed below are applicable to both structured and unstructured grids.

### 2.1 *snappyHexMesh*

A typical example of an automated grid regenerator for complex geometries is *snappyHexMesh*. To use it, the user provides a Stereolithography (STL) file of the geometry and a base mesh (typically a simple hexahedral block mesh). This utility then operates a three-stage meshing process of castellation, snapping, and boundary layer refinement. In the first step (castellation), cells are identified that are intersected by edges of the surface geometry; these cells are then refined by repeated cell splitting, with maximum and minimum levels of refinement being a definable parameter, and further surface refinement also being controllable. After this refinement process, all cells that lie ‘outside’ the desired geometric domain are deleted from the mesh. In the second, snapping step, vertices on the edge of the domain are ‘snapped’ to the STL surface, using an iterative process of mesh movement, cell refinement and face merging, again controlled by user-defined parameters such as number of iterations and specific mesh quality constraints. In the final and optional step, cell layers can be added to the surface to move the mesh away from the boundary so as to specifically refine a boundary layer. The whole process is robust and automated, but is controlled by a large number of user-specified parameters provided in advance as an input file. As with any meshing process, the user typically has to experiment with different settings to optimise the mesh. Mesh quality may ultimately be judged by the success of the resulting CFD run, but as a proxy, various mesh quality indicators, such as skewness and non-orthogonality, can more easily be evaluated.

Figure 4 shows the application of the *snappyHexMesh* utility to regenerate the geometry and mesh of the PitzDaily case (described in Sect. 1) during one of the design evaluations. Applying the ‘boundary-layer refinement’ stage of this utility requires a substantial number of input parameters. Previous experimentation as to the most influential parameters on the mesh quality (see [8]) has reduced this to a set of 7, whose definitions are provided below:



**Fig. 4** The figure on the top shows the snappyHexMesh utility applied to the generated STL geometry after the casting and snapping operation. The diagrams on the bottom show the grid distribution with and without the additional layering operation

- **resolveFeatureAngle**: Maximum level of refinement applied to cells that intersect with edges at angles exceeding this value.
- **nSmoothPatch**: Number of patch smoothing operations before a corresponding point is searched on the target surface. Smooth patches are more likely to be parallel to the target surface, making it more probable to find a matching point.
- **nRelaxIter**: Number of iterations to relax the mesh after moving points. When points are snapped to the target, the displacement propagates through the underlying layers of points that are not on the surface. By relaxing this propagation, a smoother displacement can be achieved.
- **nFeatureSnapIter**: The total number of iterations tried to snap points to the target. If insufficient quality is reached after nFeatureSnapIter iterations, the snapping is cancelled and the last state is recovered.
- **maxNonOrtho**: Non-orthogonality measures the angle between two faces of the same cell. In a grid with only rectangular cells, the value would be zero. Any deviation from this counts as non-orthogonal. High values mean there are very low angles that usually occur in a prism layer.
- **maxSkewness**: Skewness is the ratio between the largest and the smallest face angles in a cell. A value of 0 is the perfect cell and 1 is the worst. Within the OpenFOAM® dictionary, different quality constraints can be assigned to boundary cells and internal cells. In a simple geometry, the cells on the boundaries are more likely to be affected by skewness problems.
- **minVolRatio**: The ratio in cell volume between adjacent cells should not be too large—a large ratio leads to unacceptable interpolation errors.

Of course, these input parameters could also be subjected to optimisation with target measurements for the base geometry (e.g. [8]). It should be noted, however, that in

using the methodology of this contribution, snappyHexMesh can only remove sections from the base geometry—substantially reducing the exploration of the design space.

## 2.2 Grid Distortion Methods

The available techniques for modifying the grid can be separated into two groups: fixed group methods (such as the immersed boundary) and moving grid methods, with the Arbitrary Lagrangian–Eulerian (ALE) approach as a representative. In [9], the advantages of the ALE approach over the fixed-mesh alternatives is described. This analysis is based on the method’s ability to maintain a high-quality grid near the moving body, resulting in a better representation of the boundary interface in this region. In the ALE approach, the grid is moved to allow for the distortion of the boundary’s shape. This can be achieved through squeezing and stretching the surrounding cells and their associated vertices. For the FV method, the conservation equation of property,  $\phi$ , over an arbitrary moving control volume,  $V_C$ , in integral form is:

$$\frac{d}{dt} \int_{V_C} \phi dV_C + \int_A dA \cdot (\vec{u} - \vec{u}_b) \phi = \int_{V_C} \nabla \cdot (\Gamma \nabla \phi) dV_C, \quad (2)$$

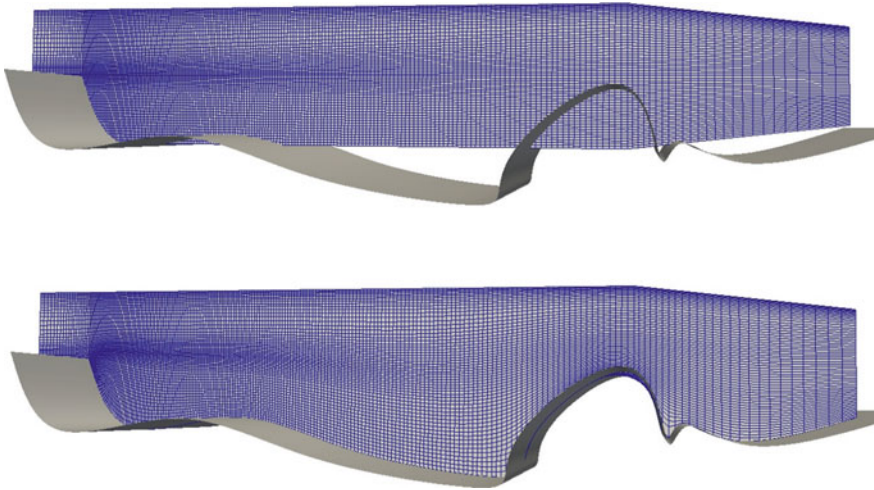
where  $\vec{u}$  is the velocity vector,  $A$  is the cell surface normal vector and  $\vec{u}_b$  is the boundary velocity vector of the cell face. To govern the vertex motion, OpenFOAM® adopts a Laplacian smoothing scheme, described by

$$\nabla \cdot (\gamma \nabla u_p) = 0, \quad (3)$$

where  $u_p$  is the point velocity, which is imposed at each vertex of the control volume. The boundary velocity  $u_b$  is interpolated from  $u_p$ . The boundary conditions for Eq. 3 are enforced from the known boundary motion, e.g. a moving wall. The vertex position at the time level  $n + 1$  is calculated by using  $u_p$

$$x^{n+1} = x^n + u_p \Delta t. \quad (4)$$

The variable  $\gamma$  prescribes the distribution of deforming cells around the moving body. Ideally, for the Laplacian approach, the cell distortion near the moving wall should be less perturbed by the motion of the body, while with increasing distance away from the wall, the cells should have greater freedom to deform. Under this concept, the quadratic diffusion model ( $\gamma = 1/l^2$ ) has been shown to present a suitable distribution of cells around the body, with  $l$  being the distance from the moving wall. A comparison between the uniform ( $\gamma = \text{uniform}$ ) and quadratic diffusion models on the grid skewness is demonstrated in [10, 11]. As the grid motion in the whole domain is governed by Eq. 3, an interface between the static and dynamic mesh regions is not



**Fig. 5** The Laplace diffusion model ( $\gamma = \textit{uniform}$ ) applied to the generated STL surface. The ‘lowerwall’ boundary was chosen to morph into the STL surface

required. Figure 5 shows the application of the uniform diffusion model applied to the PitzDaily tutorial case. It can be seen in this figure that by stretching and squeezing the cells to fit the moving boundary, the skewness and orthogonality qualities of the mesh are compromised. To overcome this, further modifications to the solver can be achieved by including the mesh refinement utilities available in *snappyHexMesh*; this has shown promising results in previous related studies (e.g. [12, 13]).

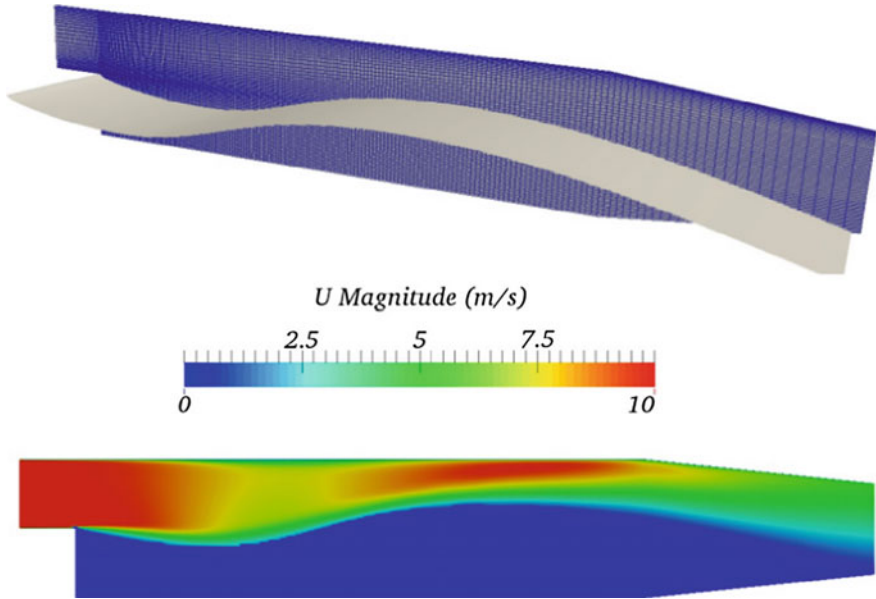
### 2.3 Immersed Boundary Method (IBM)

Immersed boundary methods provide a promising alternative to the classical body-fitted discretisations. In the Immersed Boundary approach, the CFD grid does not conform to the geometry of the object, eliminating the problem of the mesh deformation. When combined with an efficient flow solver, this approach is particularly well-suited for the automated analysis of complex geometry problems. An additional advantage of non-boundary-conforming numerical methods is that there is no need to compute the internal mesh sensitivity with a substantial reduction in user coding and computation effort. In summary, the advantages of IBM over Body-Fitted Mesh methods include:

- a substantially simplified grid distribution around the complex geometry;
- flexibility with the inclusion of a body motion due to the use of stationary, non-deforming background grids;

while the limitations of the IBM include:





**Fig. 6** The Immersed Boundary method applied to the PitzDaily tutorial with the additional STL surface as the immersed boundary

- the requirement of special techniques to simulate the boundary conditions for the immersed boundary;
- problems with grid resolution in the boundary layer region of the geometry;

with topics currently under investigation in the literature:

- IBM mimicking the equivalent body-fitted mesh solutions;
- automated mesh refinement around the Immersed Boundary surface.

Figure 6 demonstrates the application of IBM in the OpenFOAM® framework [14].<sup>1</sup> It can be seen in this figure that the Immersed Boundary has obstructed the cells below the spline from the inflow and outflow, in a similar manner as if that part of the domain had been removed.

<sup>1</sup>The Immersed Boundary Method is not available in OpenFOAM® 2.3.1, but is available in Foam-extend-3.2 and 4.

### 3 Conclusions

The grid distortion and generation techniques available in the OpenFOAM® framework appropriate for shape optimisation have been reviewed in this contribution. OpenFOAM® provides a convenient range of libraries for distorting the grid (using the Laplacian method) or grid regeneration (snappyHexMesh). The grid distortion technique is somewhat limited by the mesh quality after each design evaluation, but this can be overcome by including the mesh refinement libraries in snappyHexMesh. The grid regeneration technique is somewhat limited in regard to altering the geometry. A possible addition to both these techniques would be to apply grid sensitivity analysis (see [15])—defined as the partial derivative of the grid-point coordinates with respect to the design variable—within the OpenFOAM® framework. Furthermore, the recent development of the Immersed Boundary Method in OpenFOAM® provides a rather efficient approach to automated shape optimisation. However, like the remaining approaches, the generated solution is somewhat sensitive to the grid's resolution.

**Acknowledgements** This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant (reference number: EP/M017915/1) for the University of Exeter's College of Engineering, Mathematics, and Physical Sciences.

### References

1. Daniels, S.J. Rahat, A.A.M. Everson, R.M. Tabor, G.R. and J.E. Fieldsend. A suite of computationally expensive shape optimisation problems using computational fluid dynamics. In 15th International Conference on Parallel Problem Solving from Nature, PPSN2018, 15th International Conference on Parallel Problem Solving from Nature, PPSN2018, (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_24](https://doi.org/10.1007/978-3-319-99259-4_24)
2. Caputo, A. Pelagagge, P. and Salini, P.: Heat exchanger design based on economic optimisation. *Appl. Therm. Eng.* **28**, 1151–1159 (2008)
3. Catmull, E. Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. *Comput. Aided Des.* **10**(6), 350–355 (1978)
4. Back, T. Fogel, D.B. and Michalewicz, Z.: *Evolutionary computation 1: Basic algorithms and operators*. Taylor and Francis, Milton Park, Abingdon-on-Thames, Oxfordshire United Kingdom (2000)
5. Schmidt, J. Stoevesandt, B.: Dynamic mesh optimization based on the spring analogy. First Symposium on OpenFOAM® in Wind Energy. **2**, (2014) <https://doi.org/10.1051/itmconf/20140203001>
6. McRae, D. and Laffin, K.: Dynamic grid adaption and grid quality In: Thompson, J. Soni, B. Weatherhill, N. (eds) *Handbook of grid generation*, chap 34. CRC Press, London. United Kingdom (1999)
7. Petropoulou, S.: Industrial optimisation solutions based on OpenFOAM® technology. V European Conference on Computational Fluid Dynamics (ECCOMAS CFD 2010), Lisbon, Portugal. 14-17 June (2010).
8. Fabritius, B. Tabor, G.: Improving the quality of finite volume meshes through genetic optimisation. *Eng. Comput.* **32**, 425–440 (2016). <https://doi.org/10.1007/s00366-015-0423-0>

9. Tezduyar, T. Sathé, S. Schwaab, M. Conklin, B.: Arterial fluid mechanics modeling with the stabilized space time fluid structure interaction technique. *Int. J. Numer. Meth. Fluids.* **57**, 601–629 (2008)
10. Jasak, H. Tukovic, Z.: Automatic mesh motion for the unstructured Finite Volume Method. *Trans. of FAMENA* **30**(2), 1–20 (2006)
11. Jasak, H.: Dynamic mesh handling in OpenFOAM®. 47th AIAA Aerospace Sciences Meeting. Orlando, Florida. January (2009)
12. Jarpner, C.: Projection of a mesh on a .stl surface. Chalmers University of Technology. (2011)
13. Rypi, D. Bittnar, Z.: Generation of computational surface meshes of STL models. *J. Comp. Appl. Math.* **192**, 148–151 (2006). <https://doi.org/10.1016/j.cam.2005.04.054>
14. Jasak, H. Rigler, D. Tukovic, Z. Design and implementation of Immersed Boundary Method with discrete forcing approach for boundary conditions. In: 11th. World Congress on Computational Mechanics - WCCM XI 5th. European Congress on Computational Mechanics - ECCM V 6th European Congress on Computational Fluid Dynamics - ECFD VI, Barcelona; Spain. (2014)
15. Samareh, J.A.: Geometry and grid/mesh generation issues for CFD and CSM shape optimisation. *Opt. Eng.* **6**, 21–32 (2005)