Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXII, págs. 995-1011 (artículo completo)
Carlos G. García Garino, Aníbal E. Mirasso, Mario A. Storti, Miguel E. Tornello (Eds.)
Mendoza, Argentina, 19-22 Noviembre 2013

MODELING OF COMPRESSIBLE FLUID PROBLEMS WITH OPENFOAM USING DYNAMIC MESH TECHNOLOGY

Horacio J. Aguerre[a,b], Santiago Márquez Damián[a], Juan M. Gimenez[a] and Norberto M.Nigro[a]

[a]*Research Center for Computational Mechanics, CIMEC-UNL/CONICET, Güemes 3450, Santa Fé, Argentina, aguerrehoracio@gmail.com*

[b]*FRCU, Universidad Tecnológica Nacional, Ing. Pereyra 676, 3260 C. del Uruguay (ER), Argentina*

Keywords: Compressible Fluid Solver, OpenFOAM, Internal Combustion Engines, Mesh Dynamics

Abstract. Computational simulation of phenomena that are present in internal combustion engines is a suitable design tool for energetic efficiency optimization. OpenFOAM capabilities as computational code need to be extended to approach more complex problems that appear in the modelling of thermodynamic engine cycles. The present work reports the development and testing of a new application for OpenFOAM that enables the use of a compressible fluid solver with dynamic mesh technology. Both features are essential to model internal combustion engines. Therefore, the implementation in OpenFOAM suite of a topological mesh adaptation technique known as layering is presented. The layering technique is added to the diffusion based mesh movement among others, in the mesh dynamics capabilities of OpenFOAM. This approach is used in the compressible solver called `rhoPimpleDyMFoamMC` to solve fluid problems where the energy transfer on moving domains is carried out. The new computational tool is used to solve several problems with different boundary conditions in a closed piston-cylinder system. A detailed analysis of the accuracy and convergence of the pressure-velocity-energy coupling is achieved and the results are compared with theoretical change of state equations.

1 INTRODUCTION

Internal combustion engines (ICE) are an important challenge for the computational fluid dynamics. ICE demand a complex physics formulation that involves combustible injection and vaporization, ignition and reaction of the combustible mixture, and compressible fluid flow with energy transfer. All of this in a context of moving domains that involves different moving zones including for instance variable inlets and outlets.

This work describes the handling of cases where the domain boundary moves like a ICE piston-cylinder system with the simultaneous resolution of a compressible fluid including heat transfer. OpenFOAM®is a C++ object-oriented library for the resolution of problems in continuum mechanics (Weller et al., 1998). The structure of OpenFOAM®programming allows to solve problems of complex physics in moving domains by the advantage that the mesh dynamic implementation is independent of the mathematical model applied for the physics.

Mesh dynamic technology is necessary when the domain varies its shape in time and therefore the properties of interest are consequently affected. The domain can change due to a prescribed movement that is known by the case kinematics or applying a dynamic model which is influenced by the properties that are part of the solution, usually known as fluid-structure interaction. The mesh adaptation due to the domain change is done by a boundary adaptation that forces the internal mesh to relocate. The internal mesh action can be done principally in two different ways, one alternative is simply modifying the mesh points coordinates and the other one is changing the mesh topology which involves the change of the number of points, faces or cells or either their connectivity.

The OpenFOAM®official version offers to the user many capabilities in mesh dynamics. Principally, the OpenFOAM®mesh dynamics capabilities can be divided in two groups, mesh movement and the mesh topological change. For example, the mesh movement can be achieved solving a Laplacian equation with a constant or variable diffusivity that uses as boundary condition the domain boundary movement. In this method there is not any topological changes. Only the coordinates of the points are changed. Otherwise, an example of a mesh topological changer is the mesh refinement or unrefinement technique, where changes are performed in the topology of the mesh introducing or extracting from the mesh points, faces and cells.

In the next section, the implementation of a new technique of mesh dynamics in the official version of OpenFOAM®is presented. The new technique, called layering, performs topological modifications in the mesh and its use is appropriate for the treatment of ICE problems where the domain varies, for example, due to the movement of the piston and valves.

In the third section the solver `rhoPimpleDYMFoamMC` is introduced explaining the governing equations, the finite volume discretization and the strategy that is adopted to solve the coupled unknowns. The fourth section describes the cases that are used to test the new solver with the layering functionality and the results of the comparison with analytical evolutions are presented. The influence of the parameters of the coupling strategy in the solution accuracy is also evaluated.

Finally a summary of the work is exposed including final conclusions and a perspective of future works.

2 IMPLEMENTATION OF THE LAYERING TECHNIQUE

The layering technique is a mesh dynamics strategy where topological modifications are made on the mesh. The layering is part of what is consider in mesh dynamics technology as a mesh modifier. A mesh modifier consists of a specific combination of primitives mesh actions

that performs the proposed mesh modification. The primitives mesh actions can be defined as the fact of executing a minimal modification on the mesh topology, for example, to add a point, remove a cell or simply change any connectivity between the mesh entities.

2.1 Description of the layering technique

In the layering strategy, the boundary motion is followed through the internal mesh only by the adaptation of the adjacent boundary cells. The adjacent boundary cells change their size due to the movement of the boundary faces. The movement of the boundary faces is prescribed in the set up of the case and depending of the direction of the boundary face movement, two different actions are performed on the mesh.
If the boundary faces move in the direction of their inner normal, the adjacent boundary cells are compressed. When the adjacent boundary cells reach a minimum layers thickness tolerance, the adjacent boundary layer is removed and then the topology of the mesh changes, see Figure 1.

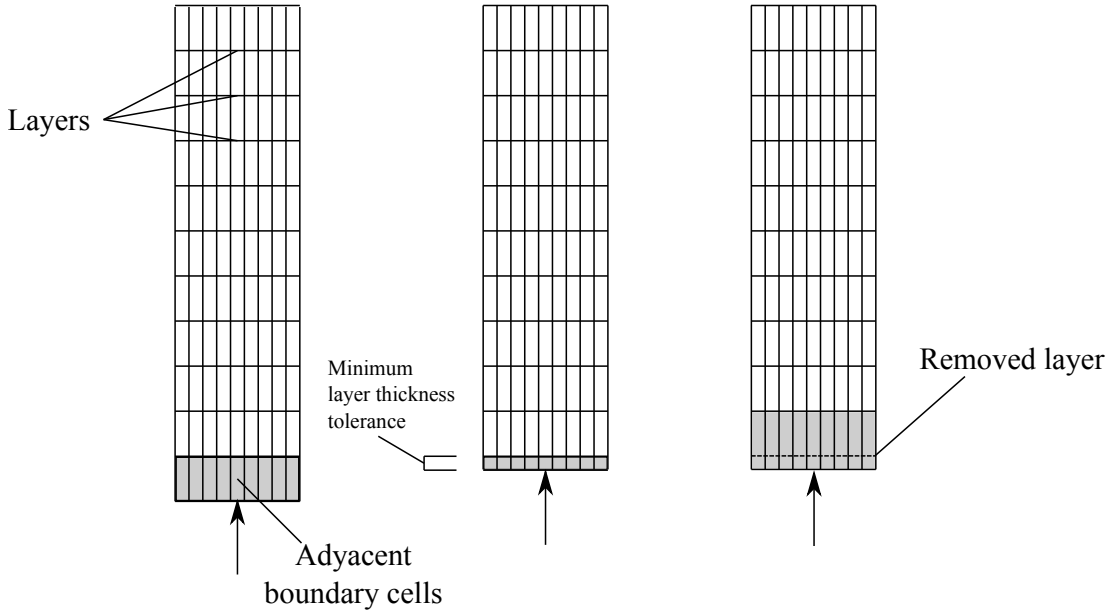

Figure 1: Removing a layer.

If the boundary faces move in the direction of their outer normal, the adjacent boundary cells are expanded. When the adjacent boundary cells reach a maximum layers thickness tolerance, a new adjacent boundary layer is added changing the topology of the mesh, see Figure 2.

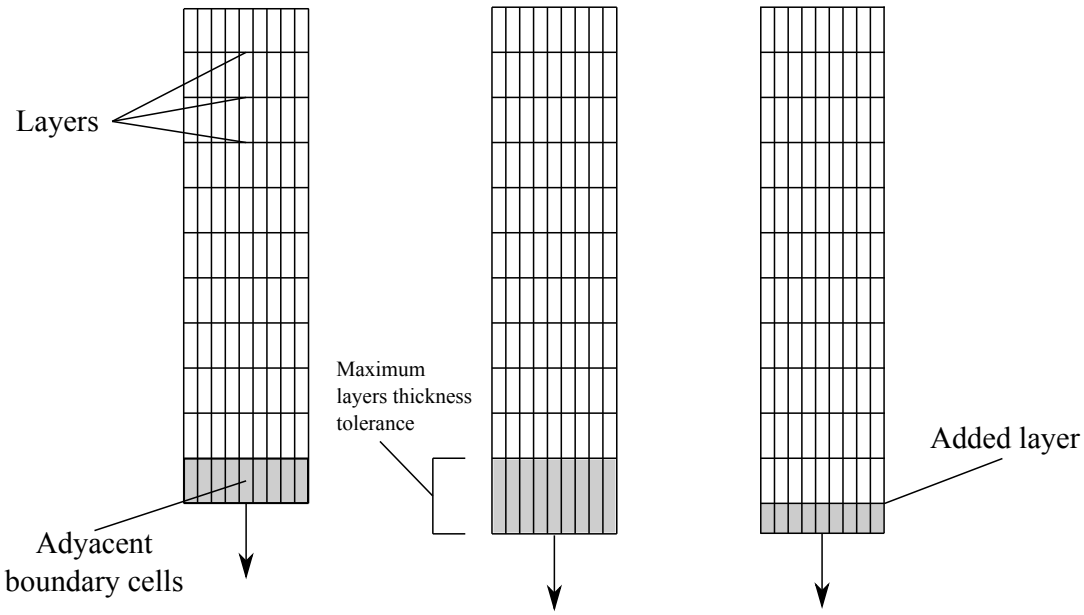

Figure 2: Adding a new layer.

2.2 Implementation of Layering in the official OpenFOAM®distribution

The implementation of the layering technique in the official version of OpenFOAM®is performed by creating user libraries without modifying the original library core. This strategy of implementation of new functionalities allows to keep the OpenFOAM®core unmodified and in this way unexpected errors are avoided.

The official version of OpenFOAM®implements the topological mesh modifiers on the library `libTopoChangerFvMesh`. The mesh modifiers methods make use of the several primitive mesh actions that are compiled on the library `libDynamicMesh`. In order to maintain the OpenFOAM®core unmodified a general mesh dynamics user library is created. The new library is named `libDynamicFvMeshFull`. In this library the implementation of the layering technique is defined on the file `layeringFvMesh.C`.

The method defined in `layeringFvMesh.C` uses the primitive mesh actions of removing or adding a cell. Both actions are implemented on the files `addCelllayer.C` and `removeCelllayer.C` respectly that are compiled on the original core library `libDynamicMesh`. The primitives mesh actions of the official version do not work properly for the desired layering technique, therefore they should be redefined. For this, a user version of the `libdynamicmesh` library that redefines some methods found on the files `addCelllayer.C` and `removeCellLayer.C` is created. The new library is named `libdynamicmeshMC`[1].

The user solvers must be compiled linking the new user libraries in order to adopt the layering functionality. The Figure 3 shows a scheme that explains the layer implementation for the solver `rhoPimpleDyMFoamMC`, indicating the libraries hierarchies and the two different frameworks, OpenFOAM®original core and the user programming.

[1]It should be noted that the postfix MC indicates that the compiled code uses Modified methods from the original Core.

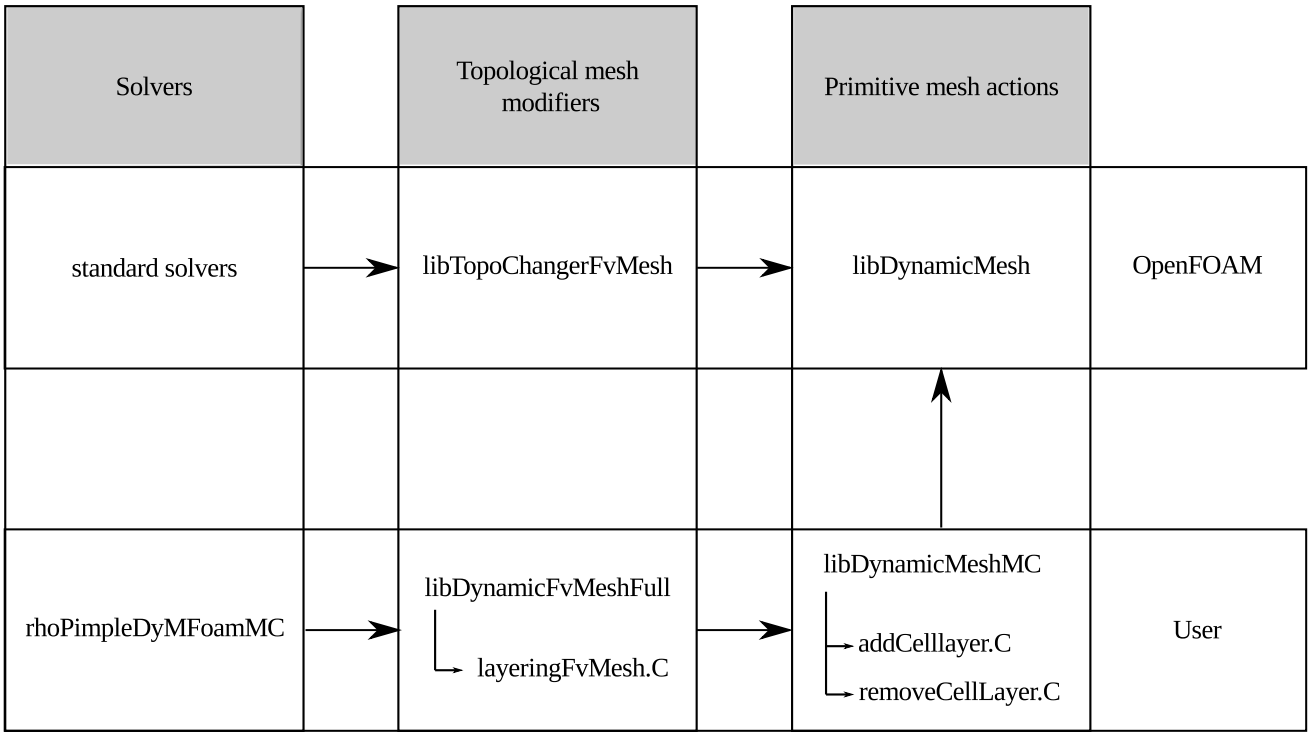

Figure 3: Layering implementation.

3 COMPRESSIBLE FLUID SOLVER

A new user application of OpenFOAM®that solves compressible fluid problems in a context of moving domains is created. The solver, called `rhoPimpleDyMFoamMC` derives from the solver `rhoPimpleDyMFoam`, that is present in the developing version of OpenFOAM®(2.2-x), and incorporates the functionality of the layering technique described in the previous section.

3.1 Mathematical model

The governing equations of the solver derive of the equation of state and the balances of mass, momentum and total energy. This equations establish a closed system for the variables, density (ρ), velocity($\mathbf{u}$), pressure (p) and internal energy ($\widehat{U}$) or enthalpy ($\widehat{h}$),

- Mass balance,

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}) \tag{1}$$

- Momentum balance,

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}\mathbf{u}) = -\boldsymbol{\nabla} p + \boldsymbol{\nabla} \cdot \boldsymbol{\tau} + \rho \mathbf{g} \tag{2}$$

- Total energy balance,

– Internal energy approach, $\widehat{U}$,

$$\begin{aligned}&\frac{\partial\left(\rho\widehat{U}\right)}{\partial t}+\boldsymbol{\nabla}\cdot\left(\rho\widehat{U}\mathbf{u}\right)+\frac{\partial\left(\rho K\right)}{\partial t}+\boldsymbol{\nabla}\cdot\left(\rho K\mathbf{u}\right)\\&=\\&-\boldsymbol{\nabla}\cdot\mathbf{q}-\boldsymbol{\nabla}\cdot\left(p\mathbf{u}\right)-\boldsymbol{\nabla}\cdot\left(\boldsymbol{\tau}\cdot\mathbf{u}\right)\end{aligned}\tag{3}$$

– Enthalpy approach, $\widehat{h}$,

$$\begin{aligned}&\frac{\partial\left(\rho\widehat{h}\right)}{\partial t}+\boldsymbol{\nabla}\cdot\left(\rho\widehat{h}\mathbf{u}\right)+\frac{\partial\left(\rho K\right)}{\partial t}+\boldsymbol{\nabla}\cdot\left(\rho K\mathbf{u}\right)-\frac{dp}{dt}\\&=\\&-\boldsymbol{\nabla}\cdot\mathbf{q}-\boldsymbol{\nabla}\cdot\left(\boldsymbol{\tau}\cdot\mathbf{u}\right)\end{aligned}\tag{4}$$

- Equation of state,

$$\rho=\frac{p}{RT}\tag{5}$$

- Constitutive relations,

$$\boldsymbol{\tau}=\mu\left[\boldsymbol{\nabla}\mathbf{u}+\left(\boldsymbol{\nabla}\mathbf{u}^{T}\right)\right]+\frac{2}{3}\mu\left(\boldsymbol{\nabla}\cdot\mathbf{u}\right)\mathbf{I}\tag{6}$$

$$\mathbf{q}=\lambda\boldsymbol{\nabla}T\tag{7}$$

where $\boldsymbol{\tau}$ is the stress tensor, g is the gravity acceleration, K is the kinematic energy, $\mathbf{q}$ is the energy flux vector, R is the ideal gas constant, μ is the dynamic viscosity, $\mathbf{I}$ is the identity tensor, and λ is the thermal conductivity.

3.2 Finite volume discretization

The following integral balance of the tensorial quantity $\boldsymbol{\psi}$ on the variable domain $\Omega(t)$ with volume $V_{(t)}$ is proposed (Jasak and Tuković, 2007),

$$\frac{d}{dt}\int_{\Omega(t)}(\rho\boldsymbol{\psi})dV+\int_{\partial\Omega(t)}\rho\boldsymbol{\psi}(\boldsymbol{u}-\boldsymbol{u}_{b})\cdot\boldsymbol{n}dA=-\int_{\partial\Omega(t)}\rho\boldsymbol{Q}_{\boldsymbol{\psi}}\cdot\boldsymbol{n}dA+\int_{\Omega(t)}\boldsymbol{S}_{\boldsymbol{\psi}}dV\tag{8}$$

where $\mathbf{u}_b$ is the moving boundary velocity, $\partial\Omega(t)$ is the domain boundary, $\mathbf{Q}_\psi$ is the surface source and $\mathbf{S}_\psi$ is the volume source .

The governing equations given by Eqn. (8) are discretized making the balance for each cell as is shown in Eqn. (9),

$$\frac{\rho^{n}\boldsymbol{\psi}^{n}V_{P}^{n}-\rho^{o}\boldsymbol{\psi}^{o}V_{P}^{o}}{\Delta t}+\sum_{f}\left(\phi_{f_{\text{rel}}}\boldsymbol{\psi}_{f}\right)\cdot\mathbf{S}_{f}=\sum_{f}\rho\left(\boldsymbol{Q}_{\psi}\cdot\mathbf{S}_{f}\right)+\mathbf{S}_{\psi}V_{P}\tag{9}$$

where the subscript P represents the cell values, $\phi_{f_{\text{rel}}}$ is the relative to mesh flux and superscripts n and o indicate the new and old time level values respectively.

The transformation between the absolute flux $\phi_{f_{\text{abs}}}$ and relative to mesh flux $\phi_{f_{\text{rel}}}$ is done by the following expression,

$$\phi_{f_{\text{rel}}} = \phi_{f_{\text{abs}}} - \phi_{f_{\text{mesh}}} \tag{10}$$

The mesh flux $\phi_{f_{\text{mesh}}}$ is calculated as the spatial flux due to the faces movement. The total flux of the faces of a cell must be balanced with the cell volume change rate (Demirdžić and Perić, 1988),

$$\begin{aligned} \phi_{f_{\text{mesh}}} &= \mathbf{u}_b \cdot \mathbf{S}_f \\ \frac{V_P^n - V_P^0}{\Delta t} &= \sum_f \phi_{f_{\text{mesh}}} \end{aligned} \tag{11}$$

3.3 Velocity-Pressure-Energy coupling

The algorithm used for the resolution of the governing equations is based on the PIMPLE method which results of combining the algorithms SIMPLE and PISO.
The SIMPLE algorithm (Patankar, 1980; Versteeg and Malalasekera, 2007) is used to solve steady-state problems where the treatment of the non-linear effects of the velocity during the resolution is more important than the precise determination of the pressure field. As each iteration is equivalent to a pseudo time step, the properties are under relaxed in order to stabilize the method and improve convergence. On the other hand, the PISO algorithm (Márquez Damián, 2013; Issa, 1986), is suitable for transient simulations where it is necessary to fully solve the velocity-pressure coupling for each time step. The non-linear effects of the velocity are reduced setting small time steps characterized by Courant numbers below one.
In transient compressible cases the error due to the non-linear effects of the velocity are more important because of the compressibility. Then the momentum equation is located in an outer loop named PIMPLE and the momentum balance can be recalculated many times as number of PIMPLE iterations.
The energy equation can be located in the PIMPLE loop or either in the PISO loop. When a thermodynamic property (temperature, density or pressure) vary rapidly in time, the energy equation should be located inside the PISO loop in order to improve the pressure temperature coupling.
The scheme represented in Figure 4 shows the PIMPLE algorithm. The total energy equation block in dotted lines indicates where this balance should be located when it is included inside the PISO loop.

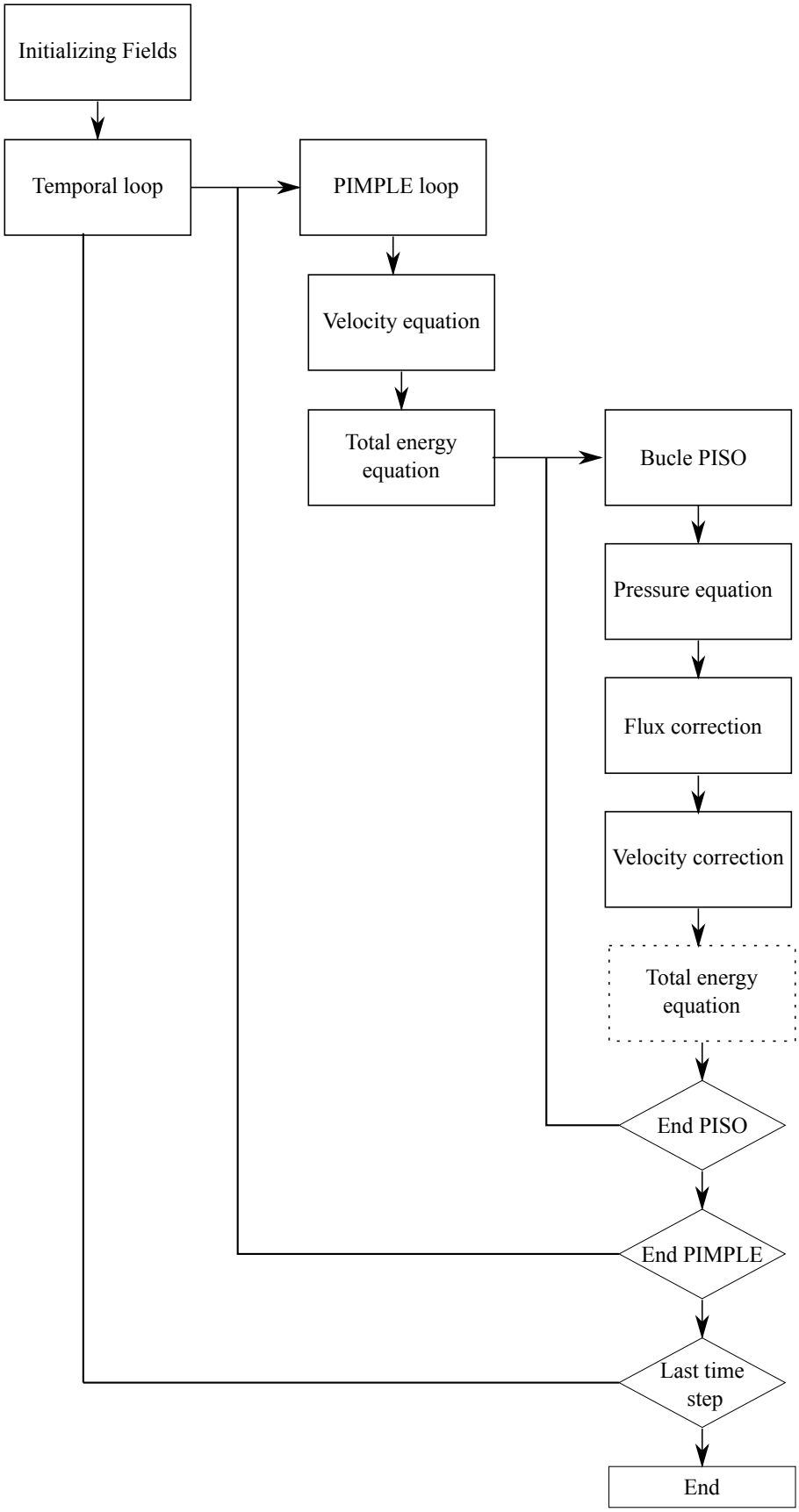

Figure 4: PIMPLE algorithm.

4 TEST CASES

In this section a validation of the solver `rhoPimpleDyMFoamMC` solving simple cases is done. The results of the numerical simulation are compared with analytical expressions.
Subsequently the velocity-pressure-energy coupling is evaluated analysing the solution accuracy for different configurations of the PIMPLE algorithm.

4.1 Cases description

The case consists of a piston-cylinder closed system. A quasi-static air compression is carried out by moving the piston with a constant velocity u_p of 1 m/s. The geometry of the model is a cylinder with an initial axial longitude L of 5 m and a cylinder diameter d of 2 m. From an initial state, $t = 0$s, the compression reaches the final state at time $t = 4$ s.
The problem consists in determining the temporal evolution of the air temperature.

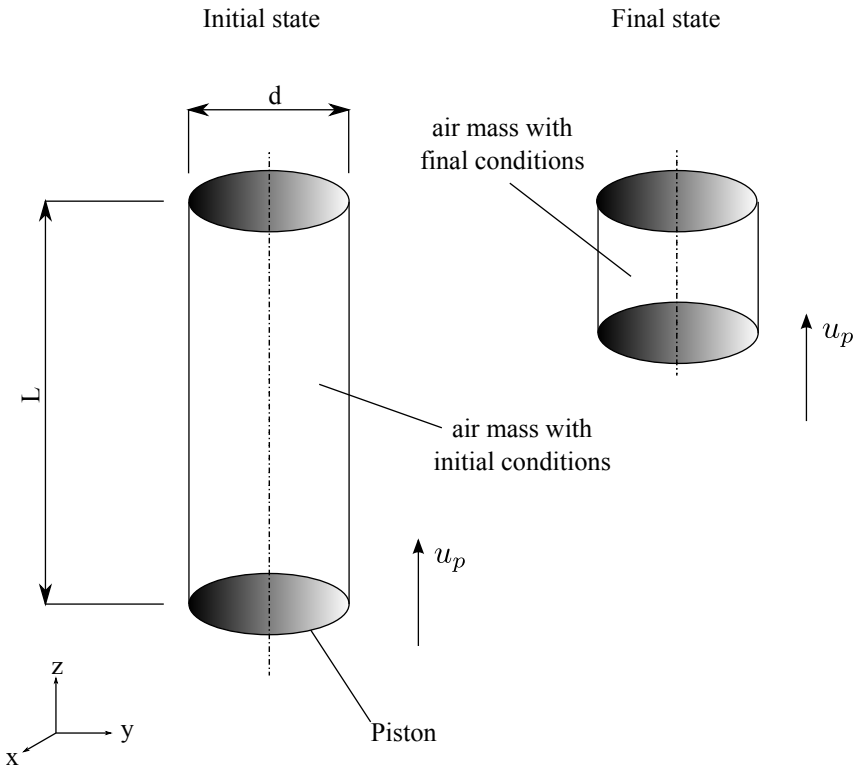

Figure 5: piston-cylinder model.

4.1.1 Boundary conditions

Two different boundary conditions for the temperature are considered. In the first case an adiabatic compression is simulated. The second case considers a Robin condition at the boundary simulating a real operating condition of ICE.

Adiabatic boundary condition The adiabatic boundary condition considers that the cylinder is isolated and therefore there is no heat flow between the ambient and the cylinder,

$$\frac{\partial T}{\partial \mathbf{n}} = 0 \qquad \text{at the boundary} \tag{12}$$

Robin boundary condition The Robin boundary condition represents that the heat flow is determined by the Newton's law of cooling,

$$\frac{\partial T}{\partial \mathbf{n}} = -h(T - T_e)A_c \qquad \text{at the boundary} \tag{13}$$

A_c is the total external area of the cylinder, T_e is the external ambient temperature and h is the pelicular coefficient.

4.1.2 Parameters

The initial conditions of the flow are those of atmospheric air at 293 K degrees[2].The parameters for the Robin boundary condition are,

$$\begin{aligned} h &= 1005 \quad \frac{\text{W}}{\text{m}^2\text{K}} \\ T_e &= 273 \quad \text{K} \end{aligned} \tag{14}$$

4.2 Deduction of the analytical expressions

In order to determine the analytical expressions, the following assumptions on the physical model are established.

- There is no dependence of the thermodynamical properties respect to the space coordinates,

$$\rho \cong \rho(t) \tag{15}$$
$$p \cong p(t) \tag{16}$$
$$T \cong T(t) \tag{17}$$

- The air velocity is zero on the whole domain,

$$\mathbf{u} = 0 \tag{18}$$

The governing equations are determined by solving the integral balance of mass, energy and space on the closed system and adding the ideal gas equation of state,

[2]the air thermal conductivity is redefined for the numerical simulation when the Robin case is considered.

$$\rho_{(t)} = \frac{M}{V_{(t)}} \qquad \text{continuity balance.} \tag{19}$$

$$V_{(t)} = \left(L_0 - \int_0^t u_{p(t)} dt\right) A_p \qquad \text{spatial balance.} \tag{20}$$

$$p_{(t)} = \rho_{(t)} R T_{(t)} \qquad \text{equation of state.} \tag{21}$$

$$C_v \frac{d\left(\rho_{(t)} V_{(t)} T_{(t)}\right)}{dt} = -h\left(T_{(t)} - T_e\right) A_{c(t)} + p_{(t)} u_{p(t)} A_p \qquad \text{energy balance.} \tag{22}$$

$$T_{(0)} = T_0 \qquad \text{temperature inital condition.} \tag{23}$$

h is equal to zero for the adiabatic case.

where $\rho(t)$ is the air density, M is the air mass; L_0 is the initial longitude of the cylinder, $V(t)$ is the volume of the cylinder, $u_p(t)$ is the piston velocity, $T(t)$ is the air temperature, $p(t)$ is the air pressure, R is the air ideal gas constant, C_v is the constant volume specific heat capacity, A_p is the piston area, A_c is the total external area of the cylinder, T_0 is the air initial temperature, h is the pelicular coefficient and T_e is the ambient temperature.

The cylinder area $A_{c(t)}$ is time dependent because the cylinder axial longitude varies,

$$\begin{aligned} A_p &= \pi r_p^2 \\ A_{c(t)} &= 2A_p + 2\pi r_p L_{(t)} \end{aligned} \tag{24}$$

r_p is the cylinder radius and $L_{(t)}$ is the cylinder axial longitude.

The system of algebraic differential equations is solved in order to obtain the thermodynamical properties $\{\rho(t), p(t), T(t)\}$.

4.3 Numerical simulation set-up

The air thermal conductivity λ should be redefined for the simulation of the Robin case. The zero dimensional model applied for the prediction of the air temperature in a closed piston-cylinder system considers that the boundary temperature is equal to the inner cylinder temperature of the air. This assumption implies to neglect the temperature gradient $\boldsymbol{\nabla} T$ inside the cylinder.
In order to compare the simulation results with the zero dimensional model, a virtual high thermal conductivity is set up for the air. The high thermal conductivity works as a temperature homogenizer and then the homogeneous domain temperature assumption given in Eqn. (17) is valid,

$$\lim_{\lambda \to \infty} \boldsymbol{\nabla} T_{(\Omega)} = 0 \tag{25}$$

4.3.1 Numerical simulation parameters

The governing equations are discretized using the following schemes:

- For the convection terms the Upwind scheme is adopted.
- The diffusion terms are solved with linear schemes.
- The temporal derivatives are interpolated by the Euler implicit scheme.

See Ferziger and Perić (2002) as reference.

The cases are solved setting a time step length of $\Delta t = 0.001$s and using 1 iteration for the PIMPLE loop and 25 iterations for the PISO loop.

4.4 Results

4.4.1 Application of the layering technique

The layering technique works successfully for the reproduction of the compression in a piston-cylinder system. In the Figure 6 the transition of the mesh from the initial state into the final state is shown.

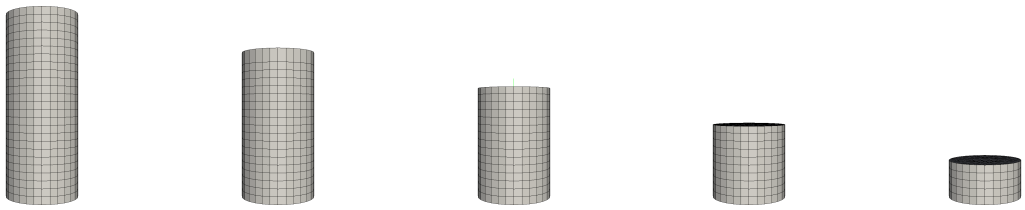

Figure 6: Layering application on a piston-cylinder system.

4.4.2 Results for the temperature temporal evolution

The results of the numerical simulations are presented as the temporal evolution of the mass weighted average of the temperature. The temperature evolution for each case is compared with the respective analytical curve.
The mass weighted average of the temperature is computed as follows,

$$T(t) \cong \frac{\sum_{n=1}^{N} (\rho_n T_n V_n)}{\sum_{n=1}^{N} (\rho_n V_n)} \tag{26}$$

The subscript n represents the value of a property on the n^{th} cell, V_n is the cell volume and N is the total number of cells of the discretized domain.

Figures 7 and 8 show the result comparisons for the adiabatic and Robin cases respectively.

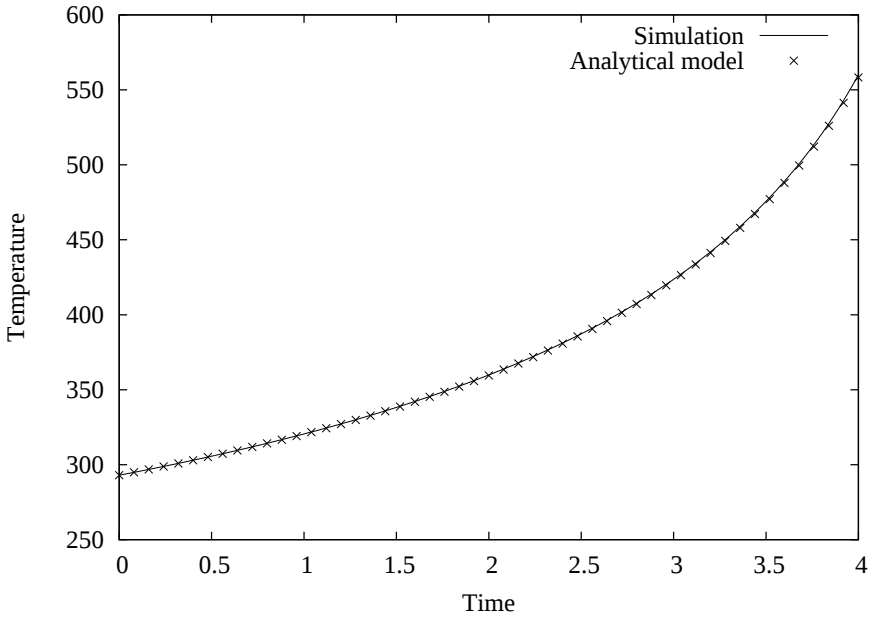

Figure 7: Temperature evolution for the adiabatic case.

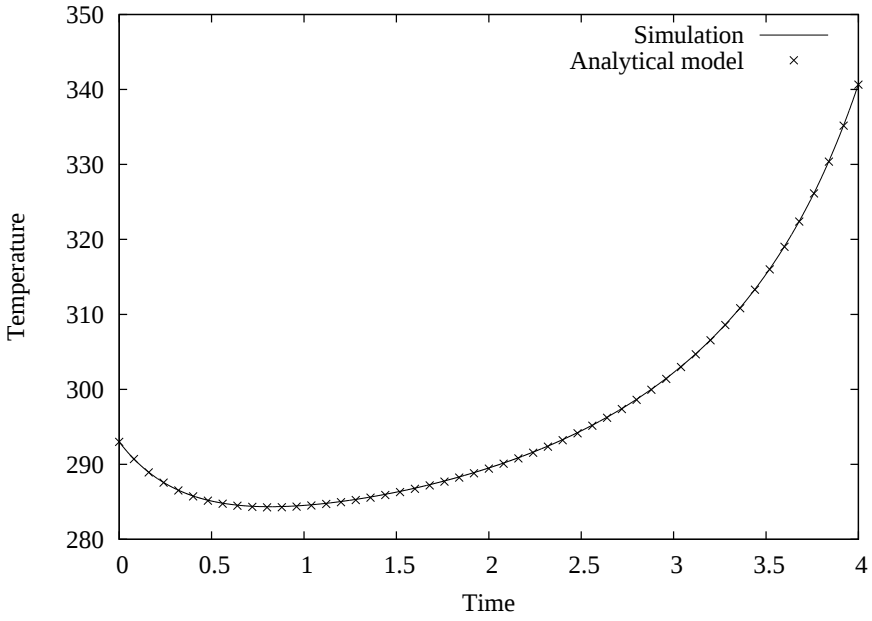

Figure 8: Temperature evolution for the Robin case.

In both cases the numerical simulation solution for the weighted mass average of the temperature fits accurately the analytical result.

4.5 PIMPLE method configuration

The accuracy of the PIMPLE method is evaluated solving the adiabatic case. The influence of the following parameters is studied,

- time step length

- numbers of iterations of the PIMPLE loop
- numbers of iterations of the PISO loop
- relaxation factors
- location of the total energy equation in the algorithm structure

The accuracy of the simulation results is determined calculating the root mean square error (RMSE) performing a comparison with the analytical solution,

$$\text{RMSE} = \sqrt{\frac{\sum_{n=0}^{N} \left[T_{s(n)} - T_{a(n\Delta t)}\right]^2}{N}} \tag{27}$$

$T_{s(n)}$ is the simulation result at time step number n, Δt is the time step length, $T_{a(n\Delta t)}$ is the analytical results evaluated at time $(n\Delta t)$ and N is the total number of time steps of the simulation.

4.5.1 Location of the total energy balance

The influence of the location of the total energy balance on the simulation results is evaluated. The time step length is fixed in $\Delta t = 0.001$s

Total energy equation located in the PISO loop,

PIMPLE iterations	PISO iterations	RMSE	Simulation time
1	3	9.46514	47s
1	10	0.51041	108s
1	25	0.50913	260s

Table 1: Total energy equation located in the PISO loop

Total energy equation located in the PIMPLE loop,

PIMPLE iterations	PISO iterations	RMSE	Simulation time
1	3	1.31571	42s
1	10	1.31570	79s
1	25	1.31570	184s

Table 2: Total energy equation located in the PIMPLE loop

The location of the total energy balance in the structure of solver algorithm affects the accuracy of the solution. When the energy equation is in the PIMPLE loop, the solution converges with a few PISO iterations. However the accuracy of the solver with the energy equation in the PISO loop is better.

4.5.2 Relaxation factors and PIMPLE iterations

The case is solved with additional PIMPLE iterations. The effect of the relaxation factors is interpreted. For these comparisons the energy equation is solved inside the PISO loop. The time step length is fixed as before in $\Delta t = 0.001$s

Results obtained without using relaxation factors,

PIMPLE iterations	PISO iterations	RMSE	Simulation time
2	3	0.70044	77s
10	3	0.50911	393s
30	3	0.50911	1124s

Table 3: Influence of the number of PIMPLE iterations without relaxing.

Results obtained using relaxation factors, which are 0.7 for the velocity and 0.3 for the energy, density and pressure,

PIMPLE iterations	PISO iterations	RMSE	Simulation time
2	3	5.39974	80s
10	3	9.03520	363s
30	3	0.29549	1171s
50	3	0.54461	2019s
70	3	0.50939	3316s

Table 4: Influence of the number of PIMPLE iterations with relaxing.

The relaxation factors for the adiabatic case do not improve the accuracy of the solution and the convergence of the method demands many more PIMPLE iterations. The case solved without relaxation factors converges and therefore there is no need to relax the variables.

4.5.3 Time step length

The accuracy of the solution is evaluated varying the time step length. The cases are solved with different configurations for the inner and outer numbers of iterations. Both configurations have a similar number of total PISO iterations. In all cases the energy balance is done inside the PISO loop.

Δt	PIMPLE iterations	PISO iterations	RMSE	Simulation time
0.001	1	10	0.51041	108s
0.001	3	3	0.51360	110s
0.001	1	25	0.50913	184s
0.001	5	5	0.50911	223s
0.01	1	10	0.86311	18s
0.01	3	3	0.86631	18s
0.01	1	25	0.86177	37s
0.01	5	5	0.86170	38s
0.1	1	10	4.70572	2s
0.1	3	3	4.70986	2s
0.1	1	25	4.70398	5s
0.1	5	5	4.70404	4s

Table 5: Influence of the time step length

The accuracy of the solutions decreases when the time step length is larger. This fact can not be improved considerably by increasing the total numbers of PISO iterations.

5 CONCLUSIONS

This work presented the implementation of a new mesh dynamics functionality for a compressible fluid solver that is part of the suite OpenFOAM®. The layering technique was introduced and a description of the implementation of the new functionality in OpenFOAM®was exposed. The compressible fluid solver `rhoPimpleDyMFoamMC` was studied making a brief description of the finite volume discretization and the velocity-energy-pressure coupling which is achieved by the PIMPLE algorithm.
The solver with the layering technique was tested working with a piston-cylinder closed system. A quasi static compression was solved using adiabatic and Robin boundary conditions for the temperature. The results of the simulations were compared with analytical predictions and the influence of the PIMPLE algorithm parameters in the accuracy of the solution was examined.
After solving the test cases it can be concluded that the mesh adaptation procedure for the reproduction of a piston-cylinder compression works successfully with the layering technique mesh dynamics technology. The numerical simulation executed by the solver `rhoPimpleDyMFoamMC` provides accurate results for the adopted test cases.
In reference to the PIMPLE method, it was determined for the test cases that a better accuracy is obtained when the total energy balance is located inside the PISO loop, in the algorithm structure. The use of relaxation factors for the variables is not necessary in this cases and its use slow down considerably the convergence of the method. The accuracy of the results depends of the time step length and it can not be improved increasing the numbers of PISO or PIMPLE iterations. The convergence of the velocity-energy coupling depends principally of the total number of PISO iterations that results of multiplying the number of iterations PIMPLE by the number of PISO iterations per PIMPLE loop.
As a future work, the layering technique implemented in this work must be improved in order to accomplish more complex problems. It is necessary an adaptive layering thickness tolerance for the treatment of small gaps and multiple moving zones with different velocities must be resolved in order to simulate problems that include valves or more than one piston.

6 ACKNOWLEDGMENTS

This work has received financial support from Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET, Argentina, grant PIP 112 201101 00331) and Universidad Nacional del Litoral (UNL, Argentina, grant CAI+D 2011 PI 501 201101 00435 and PJ 500 201101 00015)

REFERENCES

Demirdžić I. and Perić M. Space conservation law in finite volume calculations of fluid flow. *International Journal for Numerical Methods in Fluids*, 8(9):1037–1050, 1988.

Ferziger J. and Perić M. *Computational Methods for Fluid Dynamics*. Springer London, 2002. ISBN 9783540420743.

Issa R.I. Solution of implicitly discretized fluid flow equations by operator-splitting. *J. Comput. Phys.*, 62:40–65, 1986.

Jasak H. and Tuković Z. Automatic mesh motion for the unstructured finite volume method. *Transactions of FAMENA*, 30(2):1–18, 2007.

Márquez Damián S. *An Extended Mixture Model for the Simultaneous Treatment of Short and Long Scale Interfaces*. Ph.D. thesis, Facultad de Ingenieria y Ciencias Hidricas, Universidad Nacional del Litoral, 2013.

Patankar S. *Numerical Heat Transfer and Fluid Flow*. Series in computational and physical processes in mechanics and thermal sciences. Hemisphere Publishing Company, 1980. ISBN 9780891165224.

Versteeg H. and Malalasekera W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education Limited, 2007. ISBN 9780131274983.

Weller H., Tabor G., Jasak H., and Fureby C. A tensorial approach to computational continuum mechanics using object orientated techniques. *Computers in Physics*, 12(6):620 – 631, 1998.