An FAS-MultiGrid solver for steady, incompressible, viscous flow in OpenFoam®

L. Gasparini
Fondmetal Technologies, Italy

A new class has been written for OpenFoam® implementing the basic functionality required to develop non-linear FAS-multigrid solvers. Its first use was in the development of a solver for steady, incompressible, viscous flow based on the standard *simpleFoam* application. The new solver, named *MGsimpleFoam*, is briefly presented and applied to the computation of two classical 2D laminar flow benchmark cases: the square lid-driven cavity at Re=1000 and the circular cylinder in a channel at Re=20. Compared to the standard *simpleFoam* solver the performance increase is remarkable.

I. Introduction

Multigrid is well known[1] to be one of the most effective means to accelerate the numerical solution of both linear and non-linear problems. When implemented in the form of algebraic multigrid it is a powerful black-box tool for the solution of the large linear systems arising from the discretization of fluid dynamics equations; a typical example is the solution of the pressure equation, required by classical pressure-correction methods (e.g. SIMPLE) for the incompressible Navier-Stokes and continuity equations. The GAMG linear solver (or its corresponding GAMG smoother) is an example of this kind of tool which is available in the distribution of OpenFoam®. However, while accelerating the solution of the linear systems at each iteration of the SIMPLE algorithm, linear multigrid methods do nothing against the reducing speed of convergence of the non-linear equations with increasing mesh size. In fact when using linear-multigrid solvers for the solution of the linear systems the cost of a single iteration scales approximately linearly with the number of mesh cells, however the number of iterations (or time-steps) required to reach a specified level of convergence increases with mesh size; thus, total execution time increases at a much faster rate than the number of cells in the mesh.

Non-linear multigrid methods, also known as FAS-MG (Full Approximation Scheme Multi Grid), are an effective way to reduce computational time especially on medium and large meshes, by keeping the number of iterations (although more costly iterations, also called multigrid cycles) approximately constant with increasing mesh size. While similar in principle to linear-multigrid methods they are applied directly to the solution of the non-linear equations. Furthermore, their effectiveness can be combined with that of linear-multigrid methods, by using the latter in the solution of the linear systems at each multigrid iteration. Thus, looking for a significant improvement in the computational performances of *simpleFoam*, the implementation of an FAS-MG solver for the computation of steady, incompressible, viscous flow was attempted.

II. The new solver

The first step in the development of the new solver was to introduce the functionality of automatic generation of a series of coarser meshes (coarse level meshes) starting from the usual, unique, user defined grid, which represent the finest level of the multigrid hierarchy. Since such a process is inherent, although hidden to the user, in the GAMG linear solver class, it appeared that the fundamental components were already at disposal. Furthermore, the exploitation of the natural capability of OpenFoam® to deal with general polyhedral cells and the extremely powerful mesh-manipulation classes available within OpenFoam® allowed to keep the required coding to an (un)surprisingly very limited amount. As a result, a new class *FASMGagglomeration* has been written, based on the *GAMGagglomeration* class, implementing the generation of the coarse level *fvMesh*es and implementing the inter-level field transfer operators (prolongation and restriction) required by FAS-MG.

The second step was the development of the solver itself, with the following characteristics:

- FMG (Full Multi Grid) capability, at user control; which means that the solution can be started either on the finest-level grid or on any of the coarse level grids; in the latter case the solution on the coarse level is then used to initialize the next finer level and so on, until the finest-level grid is solved.
- V or W multigrid cycle, at user control, with specified number of pre- and post-smoothing sweeps on each grid level.
- SIMPLE-based smoother; which means that an iteration of the SIMPLE algorithm on a given mesh is applied as the basic smoother for the non-linear MG method.

The new solver, quite obviously named *MGsimpleFoam*, is currently only applicable to the solution of laminar flows, since the treatment of turbulence models within the MG-framework will require some (although limited and rather straightforward) extensions to the current implementation of the turbulence models themselves.

In the following sections the results obtained with the new solver in the computation of two simple and classical 2D laminar flow benchmark cases will be presented, and its performance will be compared to that of the standard *simpleFoam* solver. Note that since the two solvers apply (at least on the finest mesh, in the case of *MGsimpleFoam*) the same discretization schemes as well as the same pressure-velocity coupling scheme, the converged solutions delivered on the same mesh are exactly identical; thus, the focus of the presentation will be on the achievable reduction of execution times.

III. 2D square lid-driven cavity at Re=1000

The first test case is the laminar flow in a 2D square lid-driven cavity at Reynolds number of 1000, based on lid velocity and cavity edge length. This case is similar to the one included in the tutorials, although run at much higher Re.

Three computations were performed using both *simpleFoam* and *MGsimpleFoam* on a series of structured meshes of increasing resolution, going from 32x32 to 64x64 to 128x128 uniform cells.

Both solvers used the same space discretization schemes (in particular the Gauss linear scheme for the div(phi,U) term), the same linear solvers (BICCG for U, GAMG for p) with identical options and finally they also used the same parameters and relaxation factors for the SIMPLE scheme (no orthogonal correction iterations, relaxation factor equal to 0.3 for p, 0.7 for U).

In *MGsimpleFoam* coarse level grids where automatically generated using:

```
nCellsInCoarsestLevel   50
agglomerator            faceAreaPair
mergeLevels             2
```

and a V-cycle multigrid sequence with 1 pre- & post-smoothing sweep on the finest grid and 2 pre- & post-smoothing sweeps on coarser grids was adopted (without many optimization efforts). Although, as mentioned, the code has FMG capability, the solution was started directly on the finest level mesh to provide a straightforward comparison with *simpleFoam*.

Figure 1 shows the convergence histories of the pressure equation for the two solvers on the three different grids. It is very clear that while *simpleFoam* requires a largely increasing number of iterations with increasing mesh size, *MGsimpleFoam* achieves the goal of requiring a practically mesh-size-independent number of iterations to converge to machine-epsilon.

Figure 2 shows that although the cost of each iteration (i.e. of each multigrid cycle) in *MGsimpleFoam* is approximately 2.5 times higher, there is a significant and growing improvement in computational time, with the new solver being more than 10 times faster on the finest mesh. Note that the reported computational time includes the time spent for the initial generation of all coarse level meshes in *MGsimpleFoam*.

As mentioned above, converged solutions are identical for both solvers as shown in figure 3 in the case of the 128x128 cells mesh.

IV. 2D laminar cylinder in a channel at Re=20

The second test case presented is the steady laminar flow benchmark of Schäfer and Turek[2]: it consists in a circular cylinder in a channel at Re=20, based on average inflow velocity and cylinder diameter. It has been solved on two unstructured meshes having approximately 15000 (coarse) and 60000 (fine) triangular cells.

Space discretization schemes, linear solvers and SIMPLE parameters are identical to the previous case and again identical for both solvers. Now however, due to the irregular unstructured mesh, one non orthogonal correction iteration is specified. Furthermore, the set of options used in *MGsimpleFoam* for the generation of coarse level meshes now reads:

```
nCellsInCoarsestLevel   100
agglomerator            faceAreaPair
mergeLevels             1
```

The specification of a `mergeLevels` equal to 1 results in the generation of many more coarse grid levels (6 on the coarse mesh and up to 8 on the fine mesh), and could be considered a kind of semi-coarsening methodology. On one side this increases memory occupation and the cost of each multigrid cycle, but on the other side it improves the performance of FAS-MG. It can be roughly estimated that in this case memory occupation is approximately doubled (since each coarser level mesh is approximately half the size of the finer level one) and that one *MGsimpleFoam* iteration will cost about 6-7 times more than one *simpleFoam* iteration. Finally, the same V-cycle is used as specified previously and again the solution is started directly on the finest-

level.

Figure 4 shows the user-defined mesh for the coarse grid case of 15K cells and the automatically agglomerated coarse-level meshes on level 2, 4 and 6 (coarsest), while figure 5 presents the final converged solution computed on the fine grid of 60K cells.

Figure 6 shows, again, the convergence histories of the pressure equation for the two solvers on the two different grids. It is once more clear that while *simpleFoam* requires a large number of iterations, which increases with mesh size (to the point that convergence to machine-epsilon was not reached on the finest grid after 8000 iterations), *MGsimpleFoam* always converged to machine-epsilon in less than 60 cycles.

Figure 7 shows that even in terms of computational time *MGsimpleFoam* is really much faster on both grids.

From an engineering point of view one might be interested more in the convergence of some integral quantities, like Cl and Cd, rather than on the convergence of the residual. One advantage of multigrid solutions is that the flowfield stabilizes quite quickly so that the convergence of the force coefficients is also very fast. This behavior is shown in figures 8 and 9. Here, the magnitude of the difference between the current value of the drag coefficient at each iteration and the final converged value (written with 9 digits accuracy) is plotted respectively versus the number of iterations and the execution time for both *simpleFoam* and *MGsimpleFoam*. We can see that using *MGsimpleFoam* the value of the drag coefficient is computed to 4-digit accuracy after as few as 14 iterations on both coarse and fine grids, whereas more than 1000 iterations are required by *simpleFoam* on the fine grid. This results in one order of magnitude reduction of computational time.

V. Conclusion

A new solver, named *MGsimpleFoam*, has been developed for the computation of steady, incompressible viscous flows; it applies the SIMPLE algorithm as smoother within the context of a non-linear FAS-MG method for the solution of the incompressible Navier-Stokes and continuity equations on general polyhedral meshes.

When compared to the standard *simpleFoam* solver on two simple 2D laminar flow benchmark cases, the new solver showed the expected large reduction in the number of iterations required to reach convergence and thus a really substantial reduction of computational time, up to a factor of 10 on finer meshes.

The solver is based on a new class, *FASMGagglomeration*, which takes care of the initial automatic generation of coarse level grids and of the transfer of fields between grid levels. Thus, a similar FAS-MG reformulation of most current single-grid solvers available within OpenFoam® should be feasible with limited efforts, including unsteady solvers (like *icoFoam* and *turbFoam) by* using a dual-time approach and compressible flow solvers (like *rhoCentralFoam* or the recently improved[3] original *centralFoam* solver) by using explicit Runge-Kutta smoother.

References

[1] Wesseling, P., “Introduction To Multigrid Methods”, *ICASE report*, No. 95-11, 1995.

[2] Schäfer, M. and Turek, S., “Benchmark Computations of Laminar Flow Around a Cylinder”, *Proc. DFG Priority Research Program 'Flow Simulation on High Performance Computers'* , 1996, Vieweg.

[3] http://openfoamwiki.net/index.php/TestLucaG.

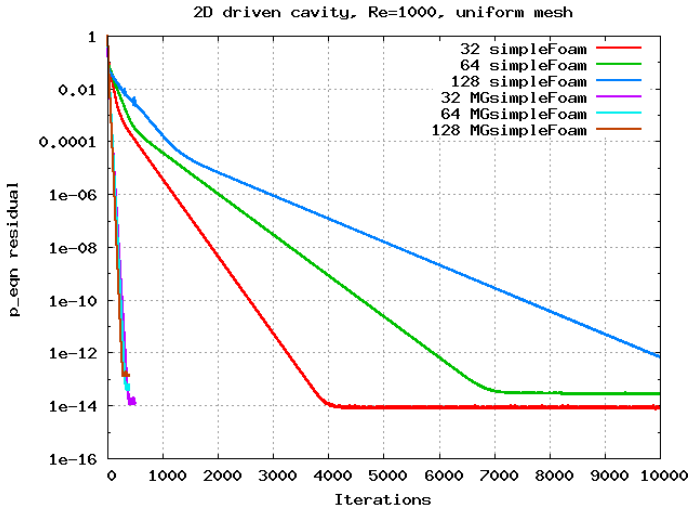


Figure 1. Pressure equation convergence histories versus number of iterations for the driven cavity test case.

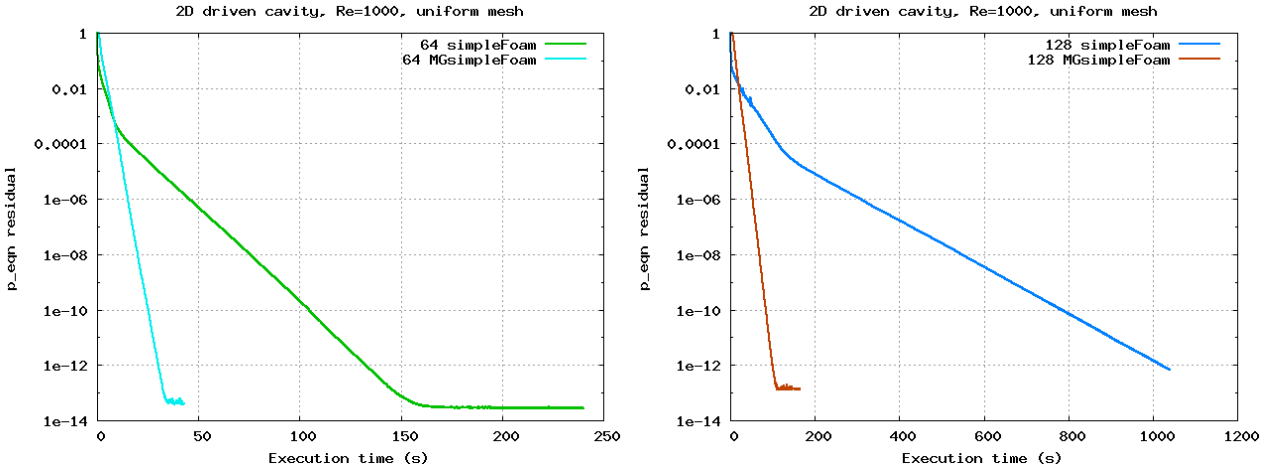


Figure 2. Pressure equation convergence histories versus execution time for the driven cavity test case.

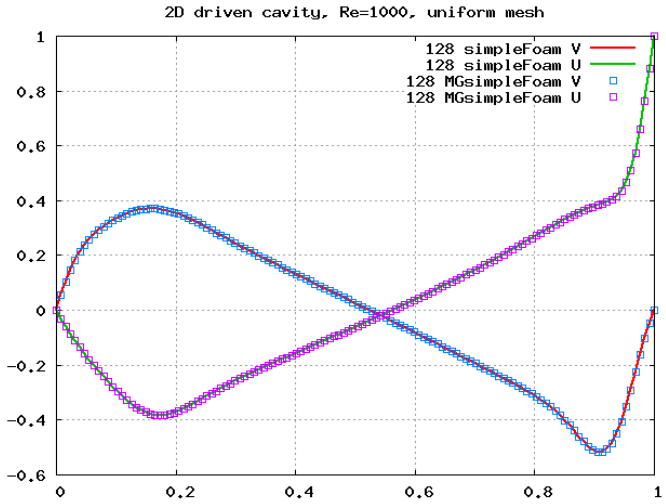
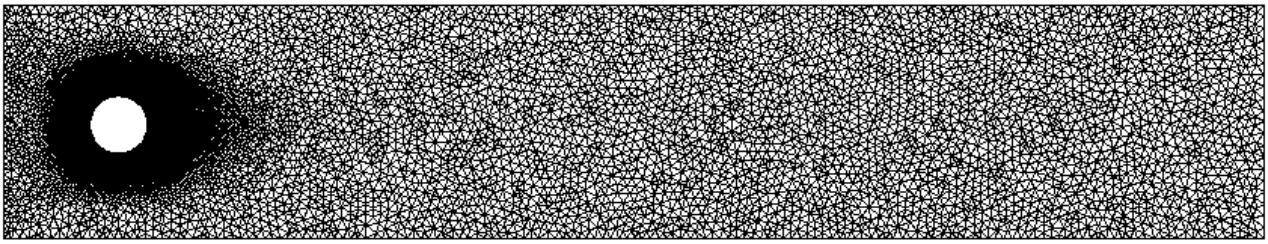
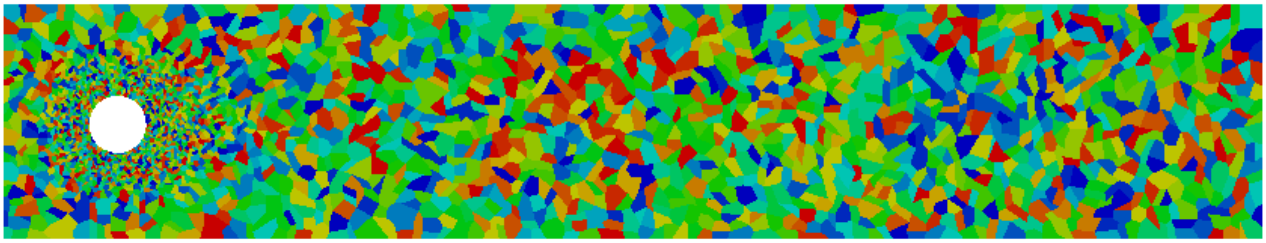


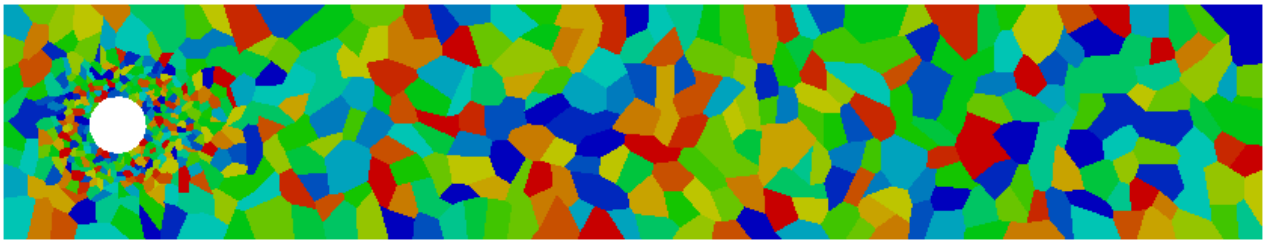
Figure 3. Solution of the driven cavity test case on the 128x128 cells mesh.



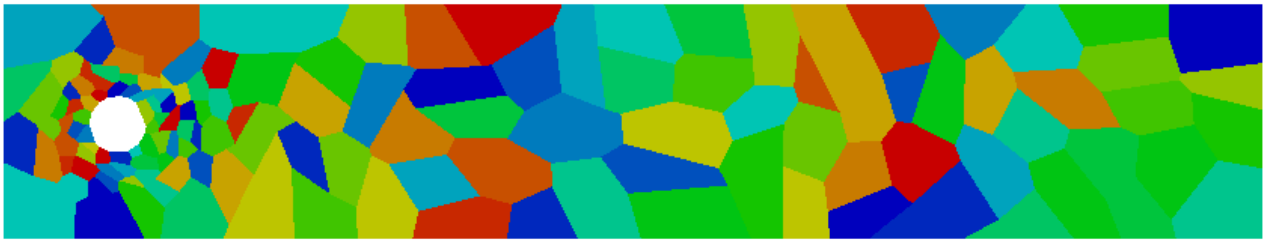
a) *Input mesh, approx. 15000 triangles.*



b) *Level-2 mesh*



c) *Level-4 mesh*



d) *Level-6 mesh*

Figure 4. Initial mesh and a few coarse level agglomerated meshes for the coarse grid laminar cylinder test case.

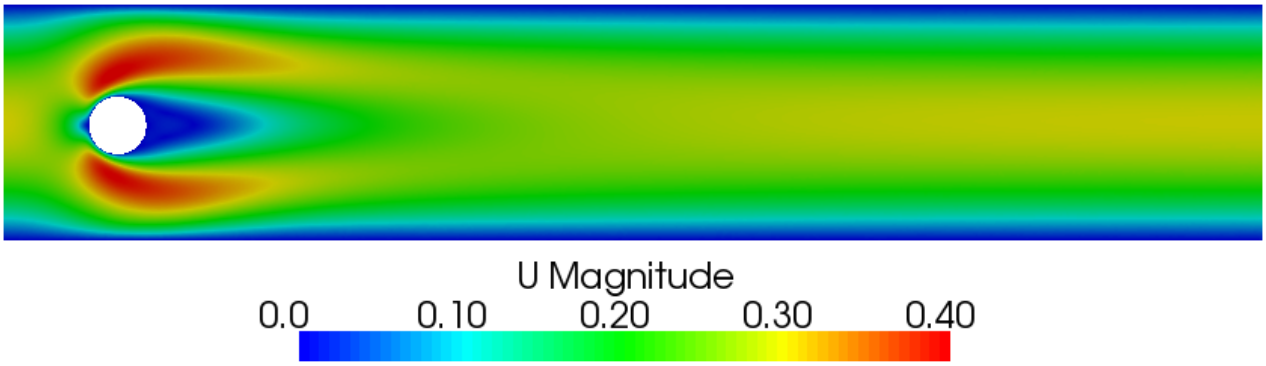


Figure 5. Converged solution on the fine grid for the laminar cylinder test case.

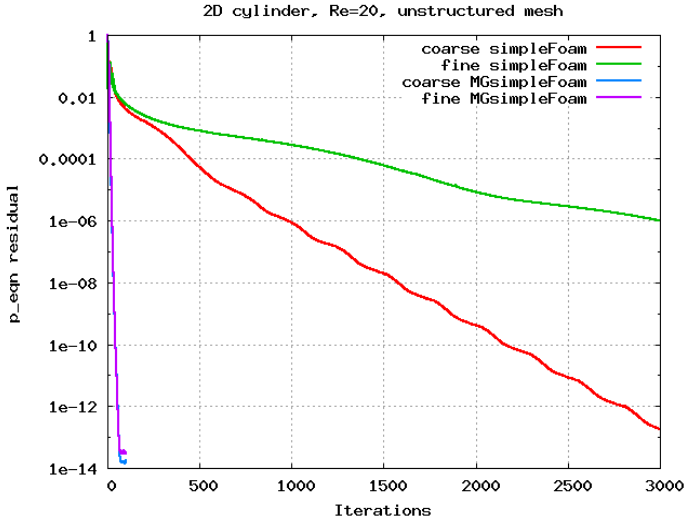


Figure 6. Pressure equation convergence histories versus number of iterations for the laminar cylinder test case.

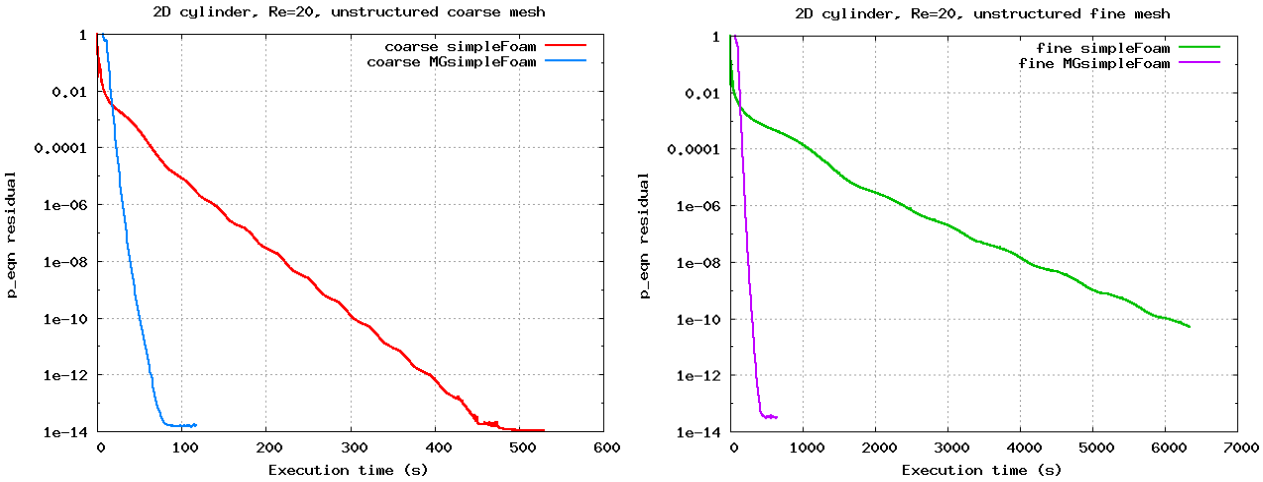


Figure 7. Pressure equation convergence histories versus execution time for the laminar cylinder test case.

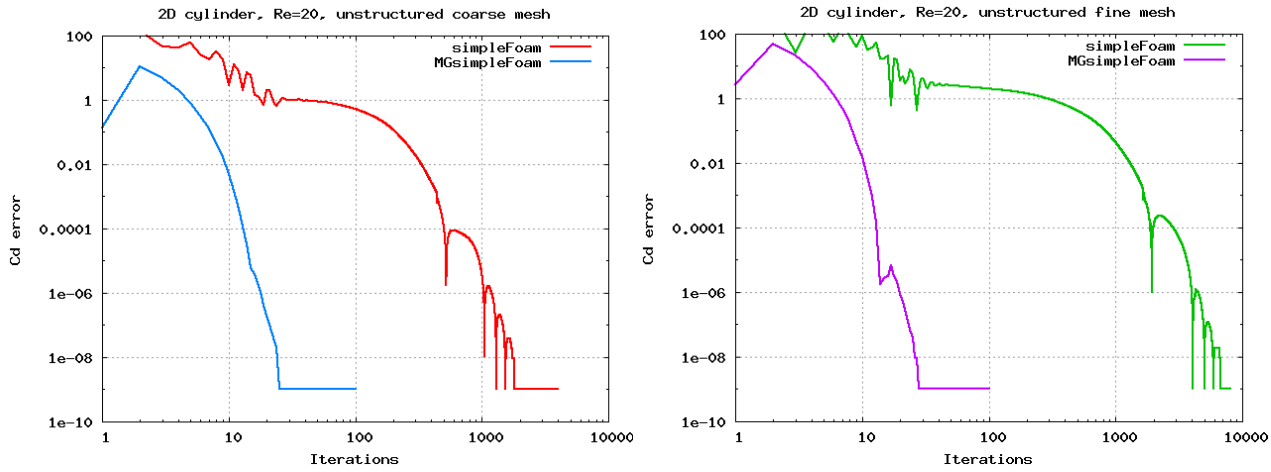


Figure 8. Drag coefficient convergence histories versus number of iterations for the laminar cylinder test case.

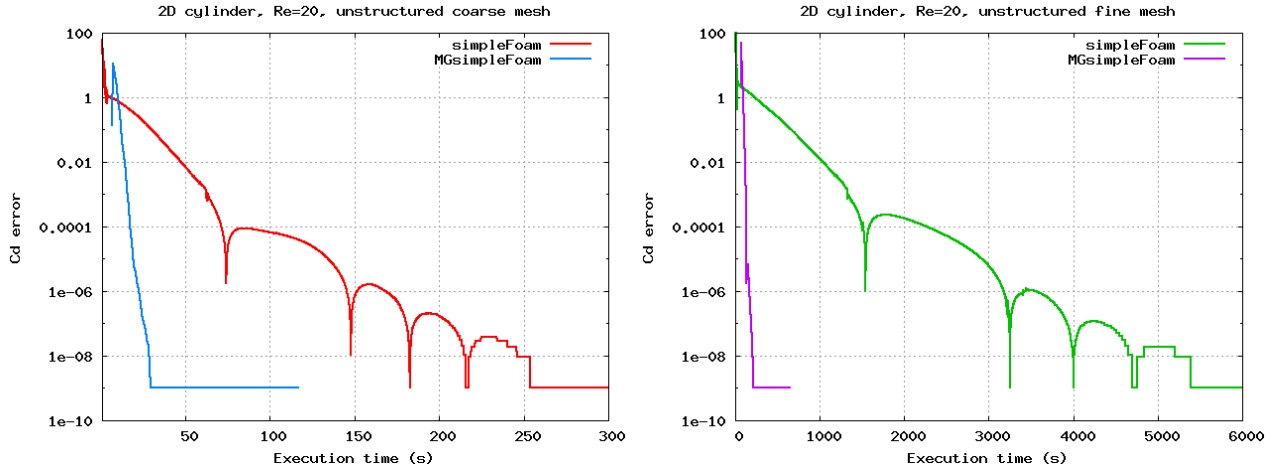


Figure 9. Drag coefficient convergence histories versus execution time for the laminar cylinder test case.