

An efficient cell vertex method for unstructured tetrahedral grids

Paul Crumpton

This paper introduces an efficient Euler solver for large tetrahedral grids (6.7×10^6 cells). The underlying method is a cell vertex finite volume formulation with a Lax–Wendroff pseudo time–stepping update. This is used as a smoother in an unstructured multigrid method, which requires a sequence of non–nested tetrahedral grids. Fast solutions are achieved by using a distributed memory parallel computer.

This paper was presented at the 1995 ICFD conference at Oxford, UK, and will appear in the proceedings.

The author would like to acknowledge the financial support of Rolls–Royce plc and DTI

Oxford University Computing Laboratory
Numerical Analysis Group
Wolfson Building
Parks Road
Oxford, England OX1 3QD

April, 1997

1 Introduction

This paper introduces an efficient Euler solver for large tetrahedral grids (6.7×10^6 cells). The underlying method is a cell vertex finite volume formulation with a Lax–Wendroff pseudo time–stepping update. This is used as a smoother in an unstructured multigrid method, which requires a sequence of non–nested tetrahedral grids. Fast solutions are achieved by using a distributed memory parallel computer.

In this paper emphasis is put on the cell vertex Euler solver which is described in the following section. The Multigrid method and parallel aspect of this work are then outlined and finally a result is presented for an aircraft configuration.

2 Cell vertex Euler Solver

In this work steady 3D Euler equations are approximated. The approach adopted here is the generalisation of the method developed for quadrilaterals in [5] and hexahedra in [4]. Since the iterative solver is to be used as a smoother within a multigrid procedure for complex geometries with millions of grid points, emphasis is put on the robustness and efficiency.

The notation used is that Greek subscripts will refer to cell averaged quantities and Roman subscripts to node based quantities. Lower case letters denote the analytic solution whereas capitals denotes mesh based approximate quantities. Bold face is used to denote a 3D vector.

The cell–vertex methodology of this method is now summarised. The unknowns \mathbf{U}_j are stored at the vertices of the tetrahedral mesh. Cell based residuals R_α are constructed, by integrating the Euler fluxes over each tetrahedra.

These cell residuals are then distributed to the nodes via distribution matrices $D_{\alpha,j}$ to construct nodal equations N_j

$$N_j = \sum_{\alpha \in C_j} (D_{\alpha,j} R_\alpha + A_{\alpha,j}) \quad (2.1)$$

where C_j is the set of cells containing node j and $A_{\alpha,j}$ is an additional artificial viscosity term, which is necessary to capture shocks and increase the numerical stability of the scheme. As in [5], the distribution matrices are chosen from the well known onestage Lax–Wendroff method.

Because the stability properties of a Lax Wendroff update are not sufficient to get satisfactory solutions, the artificial viscosity terms are needed. These are constructed via the usual blend of second and fourth order diffusion terms,

$$A_{\alpha,j} = \frac{V_\alpha}{\Delta t_\alpha \sum_{\beta \in C_j} V_\beta} \left(\epsilon^{(4)} (L_j(\mathbf{U}) - L_\alpha(\mathbf{U})) + \epsilon^{(2)} \Phi(L(p)) (\mathbf{U}_j - \mathbf{U}_\alpha) \right) \quad (2.2)$$

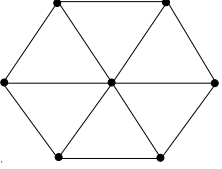
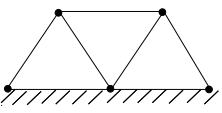
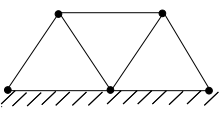
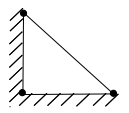
(A)		$\begin{array}{ccc} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & & -1 \\ & \frac{1}{6} & \frac{1}{6} \end{array}$	$O(h^2)u_{xx} + O(h^2)u_{yy}$
(B)		$\begin{array}{ccc} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & & -1 \\ & \frac{1}{4} & \frac{1}{4} \end{array}$	$O(h^2)u_{xx} + O(h)u_y$
(C)		$\begin{array}{ccc} & 0 & 0 \\ \frac{1}{2} & & -1 \\ & \frac{1}{2} & \frac{1}{2} \end{array}$	$O(h^2)u_{xx}$
(D)		$\begin{array}{cc} \frac{1}{2} & \\ -1 & \frac{1}{2} \end{array}$	$O(h)u_x + O(h)u_y$

Figure 1: Stencils for pseudo Laplacian

where $L(\cdot)$ is a pseudo Laplacian, p is pressure and Φ is a function of the pseudo Laplacian of p , ie. a pressure switch which is $O(1)$ near shocks and $O(h^2)$ elsewhere. $\epsilon^{(4)}, \epsilon^{(2)}$ are global constants which control the fourth and second order viscosity respectively. The cell based terms U_α and $L_\alpha(U)$ in (2.2) are evaluated by averaging over tetrahedron α . Much of this paper is devoted to the design of accurate and robust ways of evaluating the pseudo Laplacians.

A straightforward pseudo timestepping relaxation scheme can then be applied to the nodal equations, this iteration is used as a smoother within a multigrid procedure.

2.1 Evaluation of Pseudo–Laplacians

The role of the pseudo Laplacians is to maintain numerical stability without affecting the accuracy of the underlying scheme. This section describes a computationally cheap construction of such a Laplacian. The approach adopted is easily generalisable to a wide variety of schemes. The motivation for the method is the knowledge that reliable and robust convergence is obtained from $L(\cdot)$ defined as:

$$L_j(U) = \frac{1}{\#C_j} \sum_{\alpha \in C_j} (U_\alpha - U_j) \quad (2.3)$$

where U_α is an average over tetrahedra α . Similar Laplacians have been used by

[7]. It is useful to see the Taylor series expansion about node $\mathbf{x}_j = (x_j, y_j, z_j)^T$ for this Laplacian; this is presented ignoring the averaging to the cell centre; as this just confuses the issue.

$$L_j(U) = \frac{1}{\#C_j} \sum_{\alpha \in C_j} \left(\delta \mathbf{x}_{\alpha,j} \cdot \nabla u|_{\mathbf{x}_j} + \frac{1}{2} \left((\delta x_{\alpha,j})^2 u_{xx} + (\delta y_{\alpha,j})^2 u_{yy} + (\delta z_{\alpha,j})^2 u_{zz} \right) \right. \\ \left. + (\delta x_{\alpha,j} \delta y_{\alpha,j}) u_{xy} + (\delta x_{\alpha,j} \delta z_{\alpha,j}) u_{xz} + (\delta y_{\alpha,j} \delta z_{\alpha,j}) u_{yz} \right) + O(h^3)$$

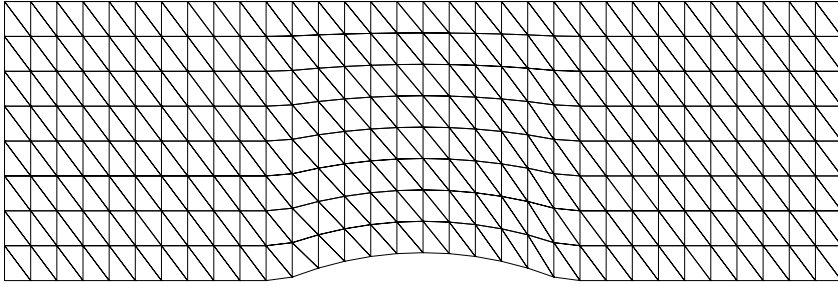
where $\delta \mathbf{x}_{\alpha,j} = \mathbf{x}_\alpha - \mathbf{x}_j$. The first term in the above expansion represents the amount the pseudo Laplacian is “sees” the linear part of the solution. The coefficients in front of the ∇u represents a pseudo Laplacian applied to the mesh co-ordinates, which will be small if the mesh is sufficiently smooth; this cannot be relied on for grids involving complex geometries. This equation reveals the motivation behind the $\frac{1}{\#C_j}$ scaling ie. to control the dissipation added from node to node. On 2D grids this scaling is less important since the majority of nodes away from the boundaries have 6 surrounding triangles. In 3D this is not the case, so this scaling removes an obvious grid dependency from $L(\cdot)$.

It is useful to look at the mesh based stencil, especially near the boundary, four examples of stencils are given in Fig. 1. On the left of this figure is a schematic of a mesh, followed by a typical stencil, followed by the a simple Taylor expansion about the central node; assuming the mesh is smooth (ie. ignoring the linear term). Clearly it is a desirable property that the pseudo Laplacian is $O(h^2)$, thus the artificial dissipation cannot destroy the $O(h^2)$ accuracy of the underlying scheme. If at a flat boundary, differences normal to the boundary are included in the Laplacian, Fig. 1(C), then an $O(h)$ term will be introduced. This can be avoided by modifying the Laplacian as in Fig. 1(C)

$$\hat{L}_j(U) = \begin{cases} \frac{1}{\#C_j} \sum_{\beta \in C_j} (U_\beta - U_j) & \text{for } \mathbf{x}_j \text{ interior} \\ \frac{1}{\#F_j} \sum_{\gamma \in C_j} (U_\gamma - U_j) & \text{for } \mathbf{x}_j \text{ on the boundary} \end{cases} \quad (2.4)$$

where F_j is the set of boundary faces connected to node j . The improvement of this modification on the solution at the boundary is clearly visible for the simple Ni’s channel given in Fig. 2, the triangular grid shown was extended into 3D prisms that are then split into tetrahedra. In this figure, as in Fig. 3 all the calculations were performed with an $\epsilon^{(4)}$ four times larger than would normally be used, so as to exaggerate the effect of the modification.

In order to ensure the Laplacian is $O(h^2)$ on distorted grids, the linear term needs to be removed from the Taylor series expansion, equation (2.4), ie. to ensure $\hat{L}_j(U) = 0$ for linear variation of U so $\hat{L}_j(\cdot)$ is “linearly transparent”. The importance of this property was studied in [6], in which edge based weights were constructed for an $L(\cdot)$ operator ie. L is formed by looping over the edges of the grid. In this work only the cell information is known. Although the edges could be introduced into the solver this is undesirable because of the extra storage



(grid)

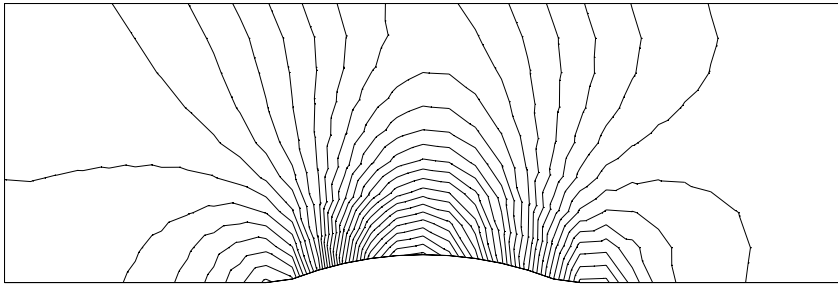
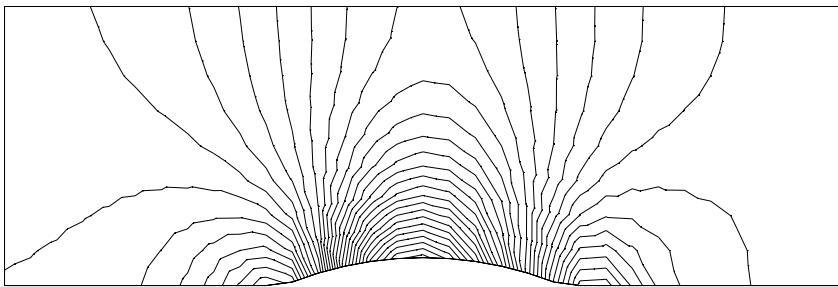
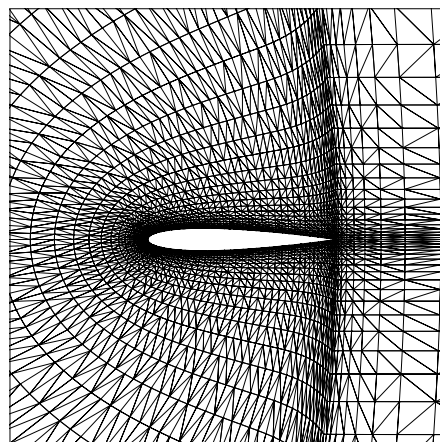
 (L)  (\hat{L})

Figure 2: Effect of boundary modification on the pseudo Laplacian



(grid)

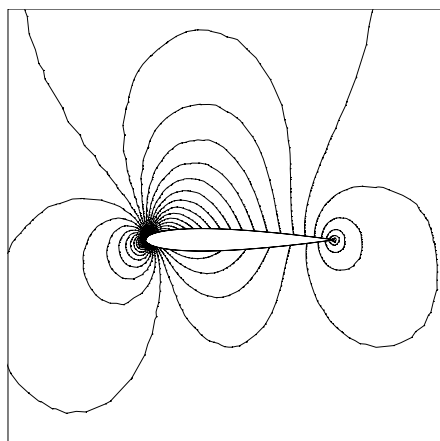
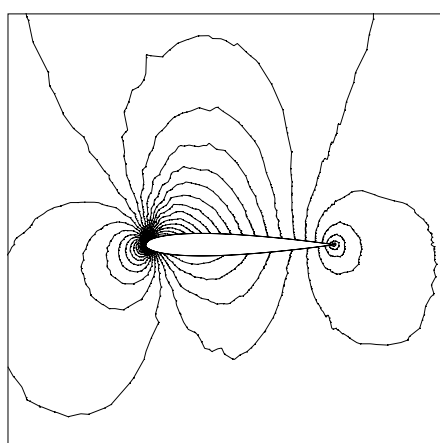
 (\hat{L}^{tt})  (\hat{L})

Figure 3: Effect of “linear transparent” pseudo Laplacian

costs. The approach adopted here is easily generalisable to a wide variety of schemes and is achieved by subtracting out the contribution of linear variation thus,

$$\hat{L}_j^{lt}(U) = \hat{L}_j(U) - (\hat{L}x_j, \hat{L}y_j, \hat{L}z_j) \cdot \nabla U_j \quad (2.5)$$

where ∇U_j is approximated by some simple differencing that is exact for linears, which in this case is

$$\nabla U_j \approx \frac{1}{\sum_{\beta \in C_j} V_\beta} \sum_{\alpha \in C_j} \mathbf{S}_{\alpha,j} \frac{1}{3} (U_k + U_l + U_m) \quad (2.6)$$

where k, l, m represent the indices of the three nodes attached to the face opposite node j in tetrahedra α . This represents a simple central difference approximation to the gradient. Since this does not damp high frequency errors (in the Fourier sense, away from boundaries), then this term will not effect the high frequency damping properties of the original $\hat{L}(\cdot)$, and consequently the robust convergence properties. Thus a “linearly transparent” modification to a pseudo Laplacian has been designed which does not effect the high frequency damping properties. This modification can easily be applied to any pseudo Laplacian.

An important feature of this modification is the little extra computational expense required. To compute the original (2.4) required a loop over tetrahedra, followed by a loop over boundary faces, with both loops accumulating to $\hat{L}_j(U)$. To construct \hat{L}_j^{lt} , the term $\mathbf{l}_j = (\hat{L}_j(x), \hat{L}_j(y), \hat{L}_j(z))^T$ needs to be stored $\forall j$, thus \hat{L}^{lt} can be accumulated using the same two loops, with the added complication of evaluating and scattering of the right hand side of equation (2.6). This requires a simple averaging of U to the faces and a dot product with $\mathbf{S}_{\alpha,j}$. Since the extra storage is based at the nodes, this overhead is small, and the compute cost is also small in comparison with the evaluation of the distribution matrices.

For many straight forward test cases (ie 2d airfoils, M6 wing, etc) the use of \hat{L}_j^{lt} proved as robust as \hat{L}_j . However, for the aircraft configuration (Fig. 4), $\hat{L}^{lt}(\cdot)$ failed to converge to machine zero, Fig. 4. This was caused by the linear transparent modification canceling $\hat{L}(\cdot)$ in corners, see Fig. 1(D), where only first differences occur (eg. where the pylon meets the leading edge of the wing). This was remedied by insisting on linear transparency only in the body co-ordinate when at a boundary. This results in a simple modification to the vector \mathbf{l}_j on the boundary, ie.

$$\mathbf{l}_j^{mod} = \mathbf{l}_j(\mathbf{x}) - (\mathbf{l} \cdot \mathbf{n}_j) \mathbf{n}_j \quad \forall j \text{ on the boundary} \quad (2.7)$$

To see what effect this has on the stencil in a corner consider Fig. 1(D). The “linear transparent” modification will destroy this stencil, however, in a corner $\hat{L}(\mathbf{x})$ and \mathbf{n}_j will be parallel, consequently $\hat{L}_j(\cdot)$ and $\hat{L}^{lt}(\cdot)$ will be identical there. This clearly remedies the convergence problem, Fig. 4. This modification allows first differences along \mathbf{n}_j to remain in the Laplacian operator. For smooth boundaries

because of the definition of $\hat{L}_j(\cdot)$ on the boundary, equation (2.4), this modification will have little effect, however, when the boundary surface changes rapidly, such as corners, dissipation is maintained through the corner, at the expense of some first difference $O(h)$ terms remaining in the Laplacian. Ultimately a compromise of accuracy has been made for robust convergence.

It is worth noting that $L_j^{lt}(\cdot)$ (ie. the “linear transparent” modification applied to equation (2.3)) automatically removes any first differencing out of L_j at the boundary, and so negates the need for the modification to \hat{L}_j . However, if equation (2.7) was used to modify \hat{L}_j at the boundary, then this first differencing at the boundary would reappear and results like those in Fig. 2 would persist.

It remains to justify the need for linear transparency. This is best exemplified by considering the solution on a randomly cross-hatched triangular mesh. Fig. 3 shows contour plots of pressure with and without the “linear transparent” modification. Clearly \hat{L}_j^{lt} has removed an error due to grid distortion.

3 Multigrid Method and Parallelisation

In 3D, where solution times can easily become excessive, the need for acceleration techniques such as multigrid is more acute than in 2D. Here, as in [8] and [7], a sequence of non-nested grids are used in a FAS multigrid procedure to accelerate the convergence of the aforementioned Euler solver. Essential requirements for this approach are: (a) Rapid mesh generation (the highly efficient method of [9] was used) and (b) An efficient grid linking algorithm (performed using directed line searches).

This multigrid application leads to a complex and intricate code. A one-off parallelisation of such a code using a message passing library such as PVM would be a time-consuming and error prone task, and the resulting code would be difficult to maintain and develop. In this case the OPlus library was used, whose purpose is to take the parallel burden from the application programmer. This has been reported in detail in [1], [3], [2].

4 Aircraft simulation

A typical surface triangulations can be seen in Fig. 4. The grid sequence used is summarised below:

grid	1	2	3	4	5
# cells	6658011	1363015	304453	65558	17748
# nodes	1063516	223392	51581	11900	3630

The finest grid of 6.7 million tetrahedra, with 1 million nodes represents a large application. Fig. 4 shows the surface pressure contours along with the comparison of single and multigrid convergence. A work unit in the convergence plot

is defined as a residual evaluation, consequently this plot demonstrates multigrid giving a considerable speed up over the single grid iteration.

This computation was performed on an eight processor IBM SP1. Each multigrid cycle took 450 seconds (wall clock); a solution to engineering accuracy (residual reduction of 5 orders of magnitude) takes less than 6 hours elapsed time. It is not possible to present the speed up over a single processor because of the memory requirement (500 Mbytes). However, good parallel efficiency has been demonstrated for smaller problems [3].

5 Conclusions

A Euler solver has been developed based on a cell-vertex discretisation and Lax-Wendroff pseudo timestepping. This requires both fourth and second order artificial diffusion. The former is devised using “linearly transparent” pseudo Laplacians. Care has been taken to ensure appropriate boundary conditions are enforced on the Laplacians, preserving the accuracy and robustness of the scheme.

A multigrid acceleration procedure along with distributed memory parallel computation is shown to be an effective strategy for solving large problems. The parallelisation of the complex FORTRAN 77 code was performed utilising a high level parallel library, OPlus.

The large aircraft calculation demonstrates how a moderately sized distributed memory parallel computer, in conjunction with then OPlus library, can deliver usable performance for large complex problems.

References

- [1] D. A. Burgess, P. I. Crumpton, and M. B. Giles. A parallel framework for unstructured mesh solvers. IFIP WG10.3 Working Conference on Programming Environments for Massively Parallel Distributed Systems, 1994.
- [2] D.A. Burgess, P.I. Crumpton, and M.B. Giles. A parallel framework for unstructured grid solvers. In S. Wagner, E.H. Hirschel, J. Périaux, and R. Piva, editors, *Computational Fluid Dynamics '94. Proceedings of the Second European Computational Fluid Dynamics Conference 5-8 September 1994 Stuttgart, Germany*, pages 391–396. John Wiley & Sons, 1994.
- [3] P. Crumpton and M.B. Giles. Aircraft computations using multigrid and an unstructured parallel library. AIAA Paper 95-0210, 1995.
- [4] P.I. Crumpton. A cell vertex method for 3d Navier Stokes solutions. Technical Report NA93/09, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD., 1993.

- [5] P.I. Crumpton, J.A. Mackenzie, and K.W. Morton. Cell vertex algorithms for the compressible Navier-Stokes equations. *J. Comput. Phys.*, 109(1):1–15, 1993.
- [6] D.G. Holmes and S.D. Connell. Solution of the 2D Navier-Stokes equations on unstructured adaptive grids. AIAA Paper 89-1932-CP, Jun 1989.
- [7] D.J. Mavriplis. Three dimensional unstructured multigrid for the Euler equations. ICASE Report No. 91-41, 1991.
- [8] J. Peraire, J. Peiro, and K. Morgan. A 3-D finite element multigrid solver for the Euler equations. AIAA Paper 92-0449, 1992.
- [9] N. P. Weatherill, O. Hassan, and D. L. Marcum. Calculation of steady compressible flowfields with the finite element method. AIAA 93-0341, 1993.

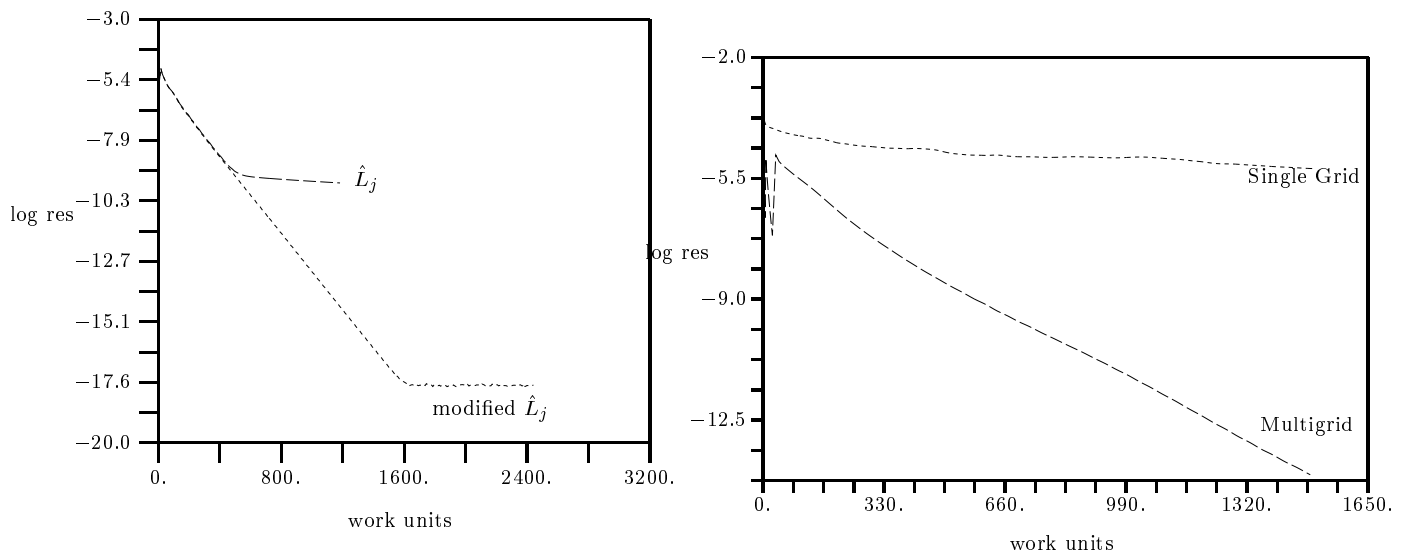
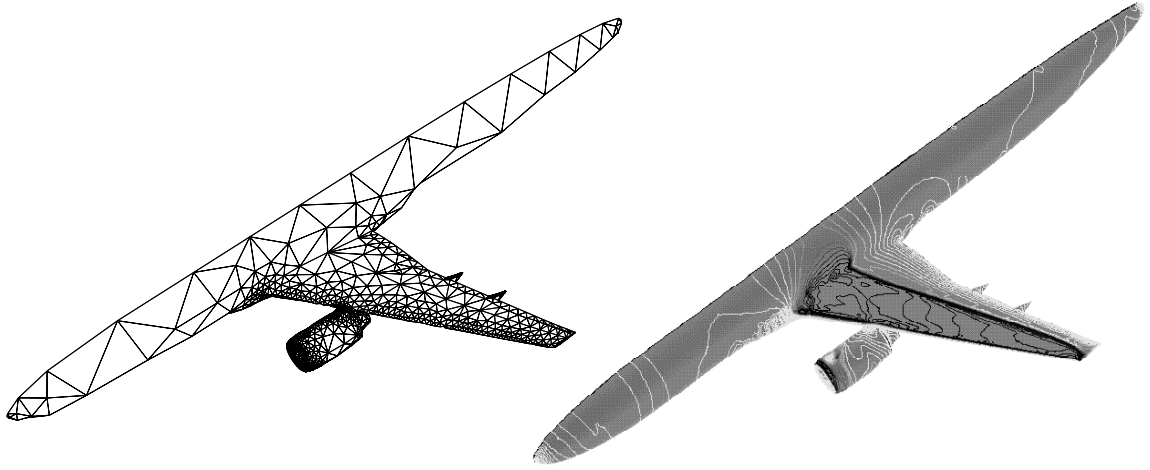


Figure 4: results for the aircraft configuration