Output Based Mesh Adaptation Using Overset Methods for Structured Meshes

Alexander W. Coppeans*, Krzysztof J. Fidkowski†, and Joaquim R.R.A. Martins‡
University of Michigan, Ann Arbor, MI, 48109

Computational fluid dynamics solvers that use structured meshes are faster per degree of freedom and often more efficient than unstructured solvers. For applications such as aerodynamic shape optimization, where hundreds of flow solution are required, structured solvers are generally preferred. However, generating structured meshes for complex geometries is a very challenging and user-intensive task. Additionally, adding local refinement to a structured mesh often requires re-meshing. Overset meshing can be used to solve both of these problems by stitching together a series of simpler meshes to conform to the geometry and to add local refinement to existing meshes. This paper presents a method of using an adjoint-weighted residual to adapt structured meshes by adding overset blocks of refinement.

I. Introduction

Computational fluid dynamics (CFD) is a powerful tool used in the engineering design process to simulate arbitrary test conditions. The complex geometries in aerospace applications provide many challenges to users especially when using structured meshes. Structured meshes are preferable due their memory efficiency compared to unstructured meshes. However, for complex geometries, it is not possible to use a single structured mesh. To get around this problem, multiblock and overset approaches can be used. Multiblock meshes combine many structured meshes in an unstructured manner to cover the flow domain, but in practice generating these meshes is a very user-intensive task. Overset meshes have been employed to ease the burden of structured mesh generation. With an implicit hole cutting method, overset mesh assembly can almost be fully automated [1].

Another drawback for structured meshes is that mesh refinement involves either global uniform refinement or re-meshing to improve local resolution. Overset meshes have been implemented as for feature-based adaptive mesh refinement [2, 3]. Feature-based adaptation has its drawbacks, as adaptation will not necessarily be added to areas of the flow that most affect engineering outputs, such as lift and drag. Shocks, for example, can be located in the wrong place due to errors in the solution upstream. Output-based methods, such as the adjoint-weighted residual method, have been shown to improve lift and drag predictions [4–7]. This suggests that using adjoint-based methods with overset meshes, mesh refinement can be efficiently added locally to structured meshes.

In this work, we developed a framework for adapting structured meshes using a coarse-space adjoint-weighted residual as an adaptive indicator. We present a method for grouping flagged cells into local refinement blocks to avoid wasted refinement and inter-grid communication. Finally, we present numerical results for a variety of inviscid test cases for the NACA 0012 airfoil. These results show that local overset mesh refinement can improve drag predictions more than global uniform refinement in a way that is non-intuitive.

II. Computational Framework

A. Computational Fluid Dynamics Solver: ADflow

The CFD we use is ADflow [8] which solves the compressible RANS and Euler equations on structured multiblock and overset meshes. For overset meshes, ADflow includes a fully automated implicit hole cutting scheme for overset connectivity. ADflow includes a discrete adjoint implementation [9] and a robust approximate Newton-Krylov start up strategy [10].

*Ph.D Candidate, Department of Aerospace Engineering.
†Professor, Aerospace Engineering Department. Associate Fellow AIAA.
‡Professor, Aerospace Engineering Department. Fellow AIAA.

B. Mesh Projection: pySurf

During uniform mesh refinement, nodes are added at the midpoint of each edge. Due to the curved nature of airfoil geometry, the new points are not necessarily on the airfoil surface. To project new nodes onto the airfoil surface, in the present work we use pySurf [11], which stores a very fine unstructured triangulated surface representation of the geometry. pySurf is able to project new nodes onto a triangulated surface, and this allows for easy extension into three-dimensions.

C. Mesh Warping: IDWarp

After nodes are projected onto the surface of the geometry, the volume mesh may need to be updated to avoid negative volumes. This is done using IDWarp [12] which uses an inverse-distance weighting method proposed in [13]. This method is advantageous as the warping preserves near-wall orthogonality, which is important for meshes used in RANS CFD. While this work does not explore adaptation with RANS, using IDWarp allows for relatively-easy extension to RANS cases.

III. Output-Based Error Estimation

The idea for output-based adaptive indicators comes from the adjoint-weighted residual, which is used to estimate how the outputs change as mesh resolution is increased. Consider states $\mathbf{U}_H$ and $\mathbf{U}_h$, which represent the primal solutions of coarse and fine-space problems with outputs of interest J_H and J_h respectively. The residuals of each of the states in their respective spaces are denoted as $\mathbf{R}_H$ and $\mathbf{R}_h$. For adjoint-based error estimation the coarse-space state is injected to the fine-space giving a state denoted by $\mathbf{U}_h^H$ where the residual in the fine-space $\mathbf{R}_h(\mathbf{U}_h^H) \neq 0$. The injected state is assumed to have the same output as the coarse-space, meaning $J_H(\mathbf{U}_H) = J_h(\mathbf{U}_h^H)$. This non-zero residual vector is then weighted with the fine-space adjoint to provide a linearized estimate of the change in output between the coarse and fine-spaces. The adjoint-weighted residual error estimate [5, 14] is then given by

$$\underbrace{J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h)}_{\delta J} \approx \boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) \tag{1}$$

The discrete adjoint vector in the fine-space, $\boldsymbol{\Psi}_h$, is the same size as the state and residual vectors in the fine-space and is defined as the solution to the linear system [5, 15]

$$\left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right)^T \boldsymbol{\Psi}_h = \left(\frac{\partial J_h}{\partial \mathbf{U}_h}\right)^T \tag{2}$$

The discrete adjoint $\boldsymbol{\Psi}_h$ gives a linearized sensitivity of the output to the residuals. A common approach for an adaptive indicator is the absolute value of the cell-wise contribution

$$\epsilon_e = |\boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)| \tag{3}$$

The cells with the largest contribution to the error are then refined to reduce the error of the simulation. This approach however, can be quite difficult, as a fine-space adjoint and a fine-space residual are needed. The fine-space adjoint solution that is used requires an additional linear solution of the adjoint equations on the fine-space, which can be costly. Instead of computing the exact fine-space adjoint, an approximate fine-space adjoint can be used where the coarse-space adjoint is injected to the fine-space in a similar manner to the primal solution. For finite-element discretizations, a fine-space can be achieved by increasing approximation order on the same mesh. For finite volume discretizations, a high-order residual can be calculated by using a high order scheme, but this often requires increasing the computational stencil for each residual calculation. Another way is to uniformly refine the mesh and to inject the converged state, and has been done for unstructured meshes [6, 7]. However, for structured overset meshes uniformly refining the mesh will change inter-grid communication, and this could lead to additional sources of error.

A. A Coarse-Space Adaptive Indicator

Due to the challenges of obtaining a fine-space residual, we instead use a coarse-space error estimate as our adaptive indicator. Instead of estimating the error from going from the coarse-space H to the fine-space h, we look at the error as

we go from the coarse-space H to an even coarse-space $\tilde{H}$. A coarse-space indicator has been used as a way to speed up mesh adaptation and shows results similar to the standard adjoint-weighted residual with much lower CPU time [16]. The coarse error estimate is given by

$$\delta J \approx \mathbf{\Psi}_H^T \mathbf{R}_H(\mathbf{U}_H^{\tilde{H}}) \tag{4}$$

where $\mathbf{U}_H^{\tilde{H}}$ is the "coarser" space solution injected into the current space. Equation 4 does not indicate how much error is in the current solution, but provides useful adaptive information. The adaptive indicator used is then written as

$$\epsilon_{e,\tilde{H}} = |\mathbf{\Psi}_H^T \mathbf{R}_H(\mathbf{U}_H^{\tilde{H}})| \tag{5}$$

Assuming that in an iterative adaptive solution, regions of high error do not experience large changes from one adaptive iteration to another, this coarse-space indicator will still provide useful information on where to adapt but in a lagged fashion. The primal solution, $\mathbf{U}_H$, is solved for using a second-order finite volume and the residual $\mathbf{R}_H(\mathbf{U}_H^{\tilde{H}})$ is evaluated using the same states but with a first-order finite volume stencil. The benefit of using this method is that no additional modifications are needed if the code has first and second order capabilities. In addition, this indicator is based on output errors rather than flow features which have been shown to waste degrees of freedom.

IV. Mesh Adaptation

In this work we implement a fixed fraction approach where a user specified percentage of cells with the highest error are flagged for refinement. Once cells are flagged for refinement, mesh resolution needs to be added. One approach would be to add one overset block, e.g. 3x3 block before refinement, on each cell. However, this would be quite costly as there would be a large amount of inter-grid communication required. Additionally, cells flagged for refinement will typically be near other cells of refinement so one block could cover multiple flagged cells. Another approach would be to find a bounding box that contains all flagged cells and add refinement there. However, this would add lots of "white-space refinement" where regions that do not have high error and are not flagged end up with unnecessary degrees of freedom. In a scenario where an airfoil has cells only around the leading edge and trailing edge flagged for refinement, adding one block around all cells would add unnecessary refinement on the upper and lower surfaces.

We would like to add refinement such that we minimize the amount of inter-grid communication while also reducing the amount of wasted degrees of freedom. To do this, we developed a greedy algorithm to group cells. First, for all cells that are flagged for refinement, we compute the average adaptive indicator around that cell using all neighbors. On cells that are not flagged for refinement the indicator is set to 0 to avoid refining cells that were not flagged. Then, the cell with the highest average error around it is chosen to start a refinement block that is initialized with all of its neighbors. We then consider growing each face of this block by one cell and compute the error that would be added if that face were grown and average it by the number of cells on that face. This average face error is then compared to the cell with the highest neighbor average error. If the face with the highest average error gained is above this threshold then that face is grown. If growing none of the faces gains more average error than this threshold then a new block is formed around that cell. Before a new block is formed, previous blocks are revisited as the cell they were compared to is now accounted for in a block, and the highest average error is now lower. Once all previous blocks are considered, a new block is formed around whatever cell has the highest average error. This process continues until every cell that has been flagged is in a block.

Figure 1 shows how the algorithm has grouped cells after 30 iterations for the mesh generated in Figure 5a.

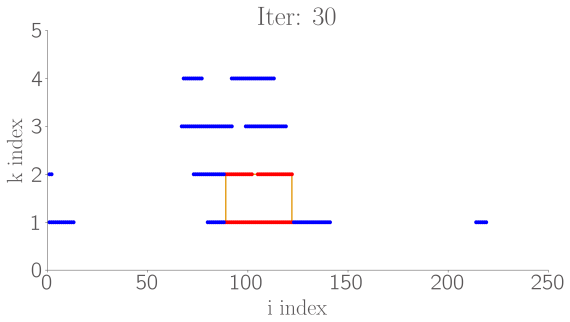


Fig. 1 Diagram of cell grouping halfway through the process.

In this figure the x axis represents the i index on the mesh which wraps around the airfoil. $i = 1$ starts at the trailing edge and as i increases it goes counter clockwise around the airfoil; $i = 219$ is the last cell by and is connected to $i = 1$. The y axis represents the k cell index in the mesh where $k = 1$ is on the airfoil wall and $k = 96$ is the cell at the farfield. The plot only shows up to $k = 5$ as only those cells were flagged for refinement.

Blue markers indicate cells that have been flagged but are not in a refinement block while red markers indicate cells that have been flagged and are grouped into a block. The orange box is the boudning box for the first refinement block and is only 2 cells high because the cell it started around a cell on the wall. Here, we consider growing the orange block in the positive and negative i directions as well as the positive k direction. We look at the error in the blue markers it would gain and average by the number of cells on that face. We then look at the cell that is not grouped with the highest average error in its neighbors and compare that to the average error the each side would pick up.

In summary, the algorithm proceeds as follows:

- Compute average error value around all flagged cells.
- Start a 3x3 block around the cell with the largest average error.
- Compute the error that would be picked up if each side of the block were grown and average it by the edge length.
- If any side has a larger average error than all the un-grouped cells' average error then grow the side that gets the most error.
- Recompute average error around cells that had neighbors added to blocks.
- Grow sides until the average error picked up by growing any side is less than any un-grouped cells average error.
- Before starting a new block, loop over all added blocks and reconsider growing each side and grow each block as much as possible.
- Start a new 3x3 block around the cell with the highest average neighbor and repeat.

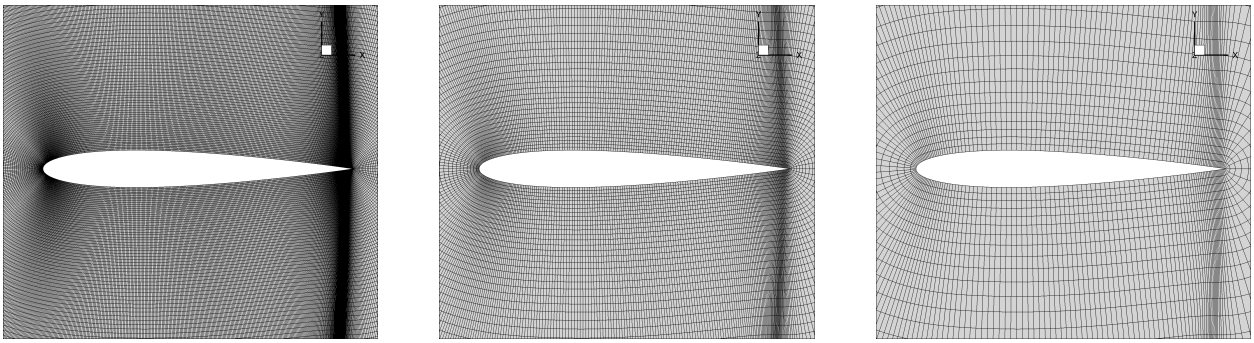
Once all the blocks have been made around the cells, the original mesh is sliced to extract each new block and uniformly refined by adding nodes at the midpoints between existing nodes. The new blocks are then all added back into the original mesh with overset boundaries except for when the wall is refined.

New nodes from the refinement are added at the midpoint of linear panels and on the wall, these are not necessarily on the airfoil surface. These new nodes need to be projected onto the airfoil surface. To do this we use pySurf contains a fine surface mesh representation of the geometry, and project nodes onto this surface. This work only studies single level refinement, so projecting points onto a surface mesh that is two-levels finer than the mesh that is being adapted ensures points stay on the exact geometry. A finer surface mesh could be used and this approach allows for easy extension into three-dimensions where exact geometric representations are much harder. Finally, once surface nodes are perturbed, we warp the volume mesh using IDWarp to ensure there are no negative volumes.

V. Numerical Results

In this section we test our adaptation algorithm on the NACA 0012 airfoil for a variety of inviscid flow conditions. These results are compared to uniform refinement. A family of three single block O-meshes with the far-field 600 chord lengths away is created by first generating the finest mesh and uniformly refining in all directions. The finest mesh created is denoted as the $L0$ mesh with $336,384$ cells. The $L1$ mesh has $84,096$ cells, and $L2$ mesh is the coarsest with $21,024$ cells. The meshes are shown in Figure 2. In each test case we add overset refinement the the $L2$ mesh. The

results show that inviscid drag estimations can be improved using overset mesh adaptation per degree of freedom.



(a) $L0$ **Mesh 876x384** **(b)** $L1$ **Mesh 438x192** **(c)** $L2$ **Mesh 219x96**

Fig. 2 Uniformly refined NACA 0012 baseline meshes.

A. $M_\infty = 0.4$ $\alpha = 2^\circ$ **NACA 0012 Airfoil**

The first test case analyzed is the NACA 0012 airfoil at 2° degree angle of attack with a free stream Mach number of $M_\infty = 0.4$. A contour plot of the coefficient of pressure on the $L0$ mesh is shown in Figure 3. The $L2$ mesh was adapted

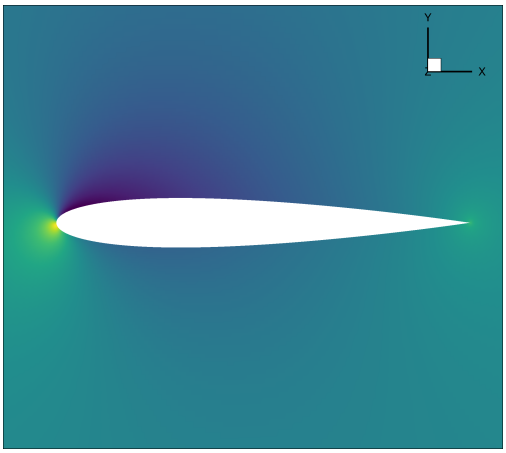
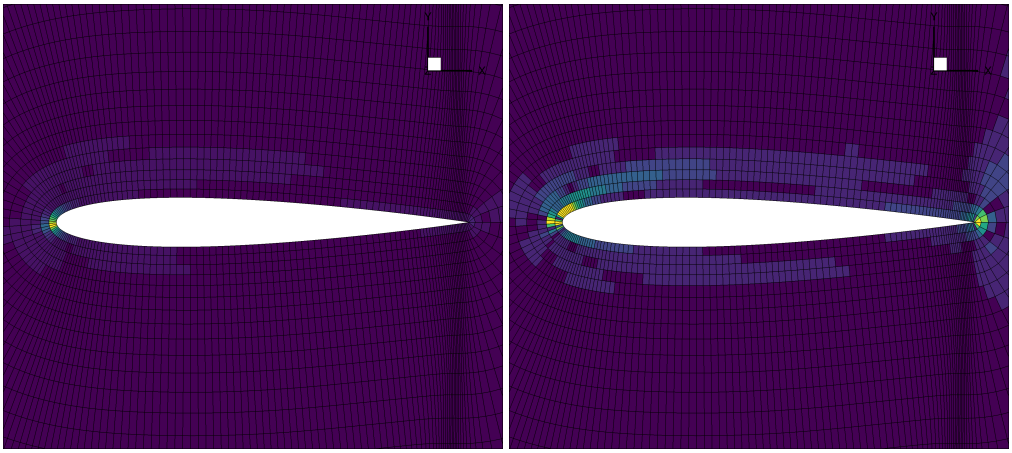


Fig. 3 Coefficient of pressure contours on the $L0$ **mesh** $M_\infty = 0.4$ $\alpha = 2^\circ$**. The color range is from -0.8 to 1.0.**

on both lift and drag outputs with a fixed fraction of 1% and 2% with only a single adaptation iteration. The adaptive error indicators on both lift and drag are shown in Figure 4.



(a) Drag adaptive indicators **(b) Lift adaptive indicators**

Fig. 4 Output-based adaptive indicators used to refine the $L2$ mesh $M_\infty = 0.4$ $\alpha = 2°$.

Both the lift and drag indicators flag the leading and trailing edges of the airfoil. However, adapting based on lift puts more emphasis on the trailing edge, while drag the highest errors occur near the stagnation point. The adapted meshes for both lift and drag are shown in Figure 5. Here we can see that the adaptation algorithm is adding refinement to the leading and trailing edges of the airfoil. This is the expected behavior for subsonic inviscid flow, where the most important parts of the flow are the stagnation streamline and where the flow leaves the airfoil. The lift and drag coefficients are plotted versus number of degrees of freedom in Figure 6. The convergence plots show that adapting the mesh improves drag predictions for all cases, getting close to the predicted 0 drag for subcritical inviscid flow. All cases except the 2% adaptation fraction based on drag error decrease the coefficient of lift relative to the baseline $L2$ mesh and move further from the lift predicted by the $L0$ mesh. Every $L2$ adapted mesh is able to predict the drag with less error than the $L1$ mesh. This is not the expected result, however, because we take the $L2$ mesh and uniformly refine only some sections of it making only some sections having at most $L1$ resolution. We would expect that adding $L1$ refinement in certain areas of the $L2$ mesh we would be able to get close to the $L1$ result but not do better. To explain this we look at the spurious entropy generation in the domain and compute the drag using Oswatitsch's drag formula given by [17]

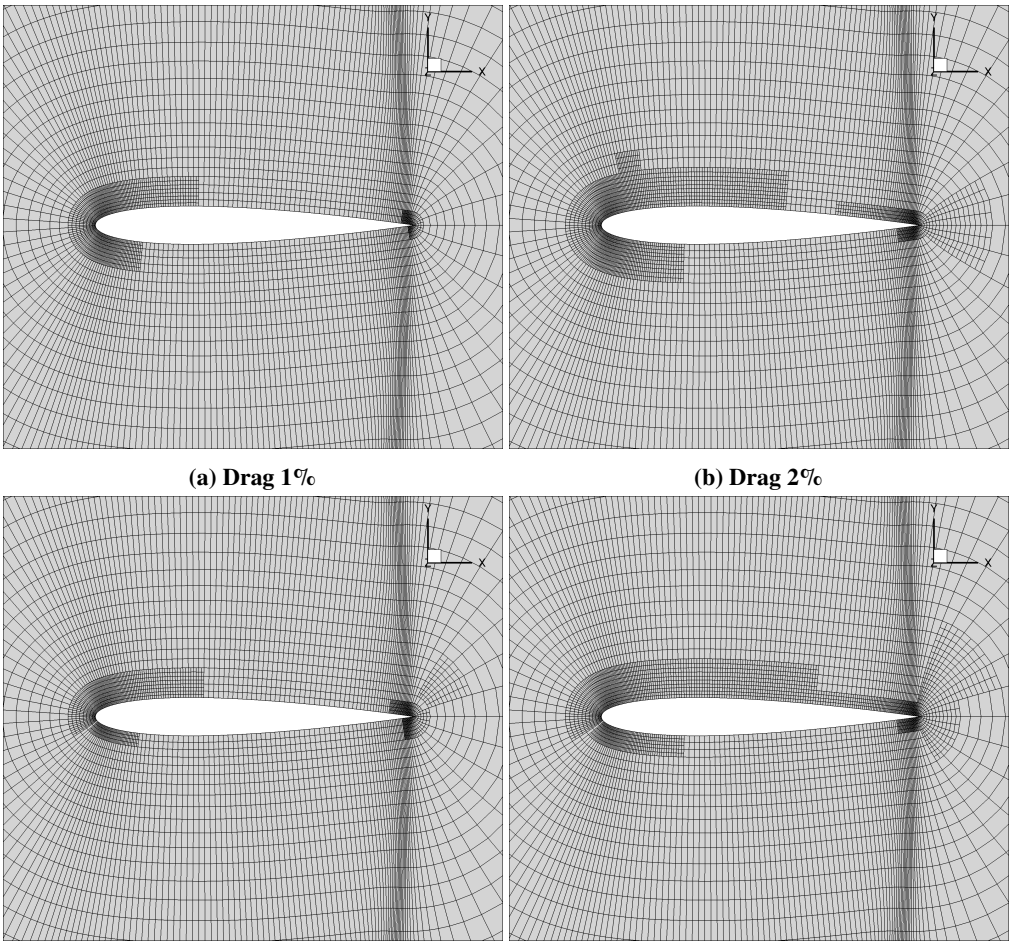
$$D_{\text{osw}} = \int_{S_\infty} u_\infty \left[1 - \sqrt{1 + \frac{2}{(\gamma - 1) M_\infty^2} (1 - e^{\frac{\Delta s}{c_p}})} \right] \rho \vec{V} \cdot \vec{n} dS \tag{6}$$

The entropy change is calculated relative to the free stream entropy.

$$\Delta s \equiv s - s_\infty, \quad s \equiv c_v \log p - c_p \log \rho$$

Equation 6 can then be normalized by $\frac{1}{2} \rho_\infty u_\infty^2 c$ to get a drag coefficient.

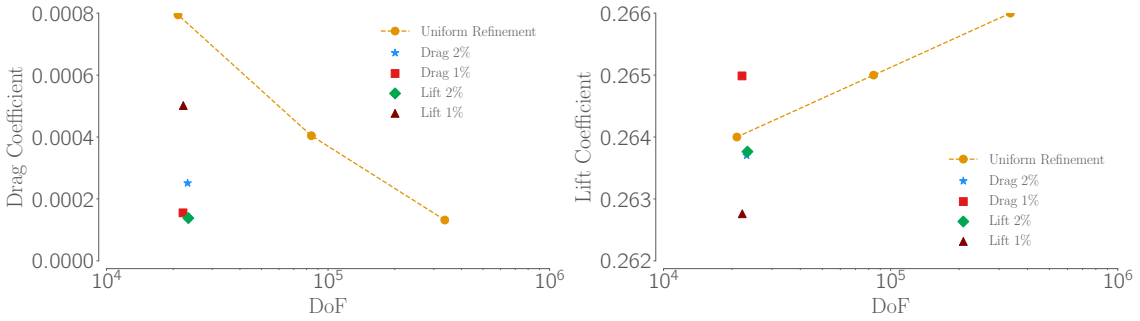
For inviscid subcritical flow any spurious entropy that is generated will show up as drag and we expect Oswatitsch's formula to be equivalent to integrating the pressure over the surface of the airfoil. Indeed for the single-block meshes, the Oswatitsch drag is very close to the pressure drag. However, once we compute the Oswatitsch's drag coefficient plotted in Figure 7 we see that for all the adapted cases that predicted lower drag than the $L1$ drag prediction, Oswatitsch's drag coefficient is negative.



(a) Drag 1% **(b) Drag 2%**

(c) Lift 1% **(d) Lift 2%**

Fig. 5 Meshes generated from adapting L2 mesh on lift and drag with 1% and 2% fixed fractions. $M_\infty = 0.4$ $\alpha = 2°$**.**



(a) Drag coefficient convergence. **(b) Lift coefficient convergence.**

Fig. 6 Output convergence for NACA 0012 Airfoil $M_\infty = 0.4$ $\alpha = 2°$**.**

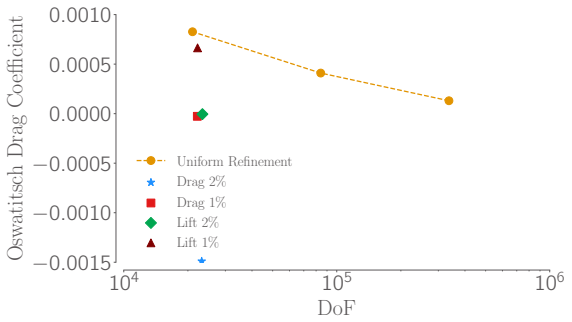
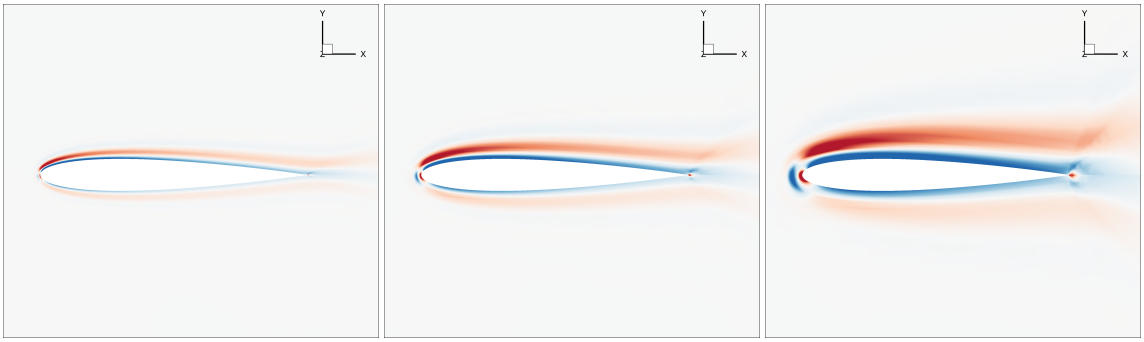
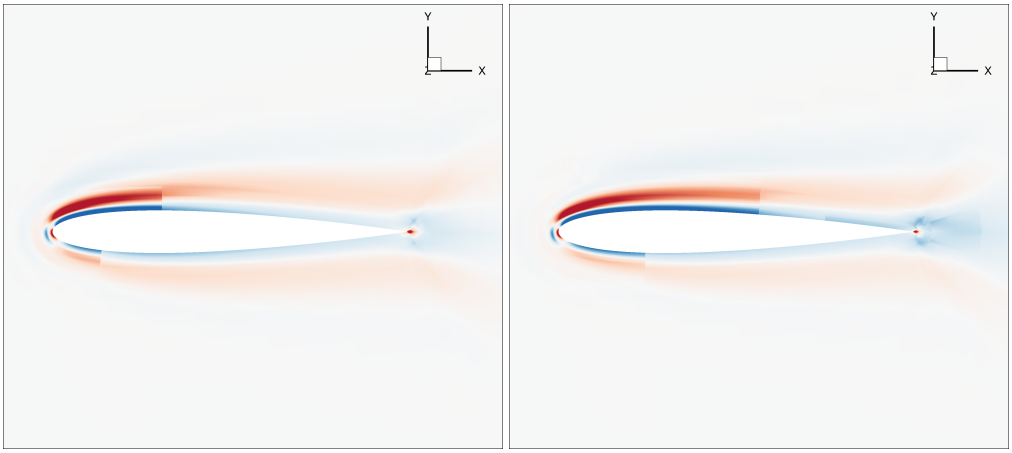


Fig. 7 Oswatitsch drag coefficient convergence $M_\infty = 0.4$ $\alpha = 2°$.

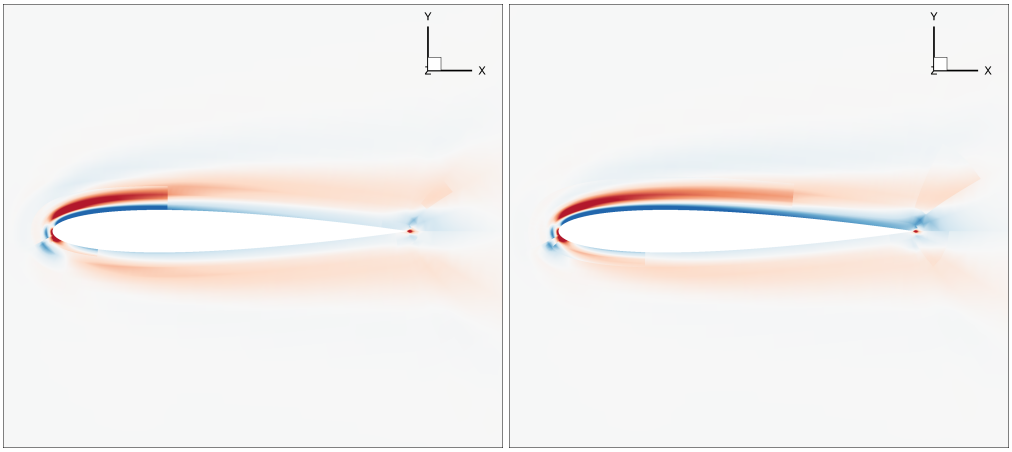
This indicates that the overset refinements blocks are removing spurious entropy from the domain. A chimera based trilinear interpolation [18] is used to transfer the solution from the background mesh to the overset refinement blocks. This interpolation is non-conservative and removes spurious entropy from the domain causing a reduction in drag. The entropy in the domain can be plotted and is shown on each mesh in Figure 8. The entropy fields show a discontinuity around the overset boundaries, indicating that at the interface, the interpolation scheme is not transferring the entropy in a conservative manner.



(a) L0. **(b) L1.** **(c) L2.**



(d) L2 adapted on drag 1%. **(e) L2 adapted on drag 2%.**



(f) L2 adapted on lift 1%. **(g) L2 adapted on lift 2%.**

Fig. 8 Entropy contours $M_\infty = 0.4$ $\alpha = 2°$.

B. $M_\infty = 0.95$ $\alpha = 0°$ NACA 0012 Airfoil

The next case tested is the NACA 0012 airfoil at 0 degrees angle of attack and at Mach number of $M_\infty = 0.95$. Mach contours are shown on the $L0$ mesh in Figure 9. The flow accelerates around the leading edge and goes supersonic on the upper and lower surfaces. As the flow leaves the airfoil, it encounters an oblique shock and remains supersonic until it reaches a terminating normal shock down stream (not shown) and reaches subsonic conditions. This test case

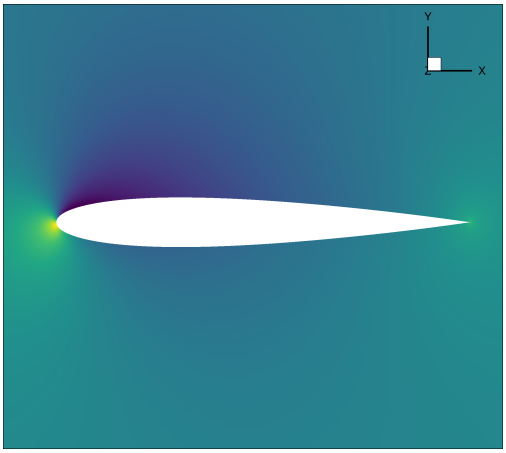
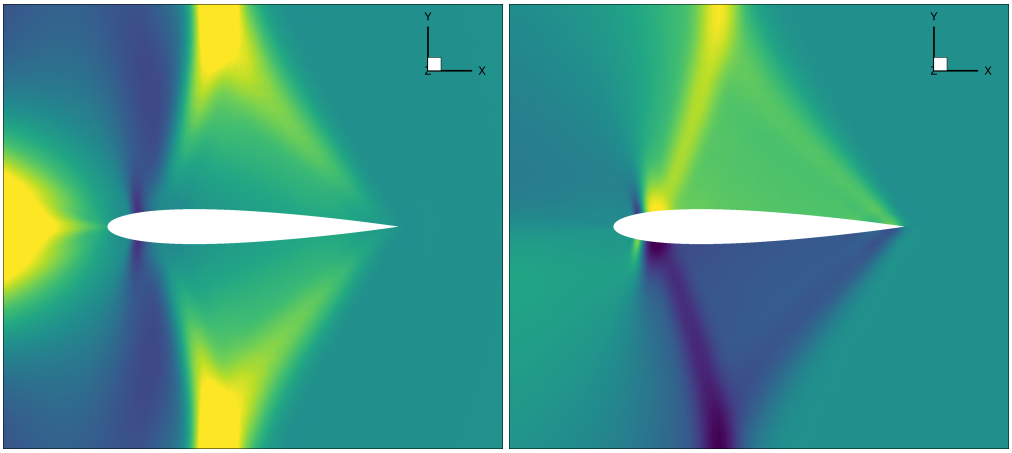


Fig. 9 Mach contours on the $L0$ mesh $M_\infty = 0.95$ $\alpha = 2°$. The color range is from 0 to 1.5.

shows the benefit of using the adjoint-weighted residual as flow down stream of the airfoil does not affect the lift and drag, so resolving any downstream flow features in unnecessary. Adapting based on flow gradients has been shown to waste degrees of freedom [6] resolving the shocks which has no benefit in improving drag calculations using a surface integration on the body. The adjoint is able to adapt only in regions where the flow affects the drag as shown in the adjoint plots in Figure 10. The adjoint variables down stream of the airfoil are zero as expected and non-zero where changing the flow will affect the output.

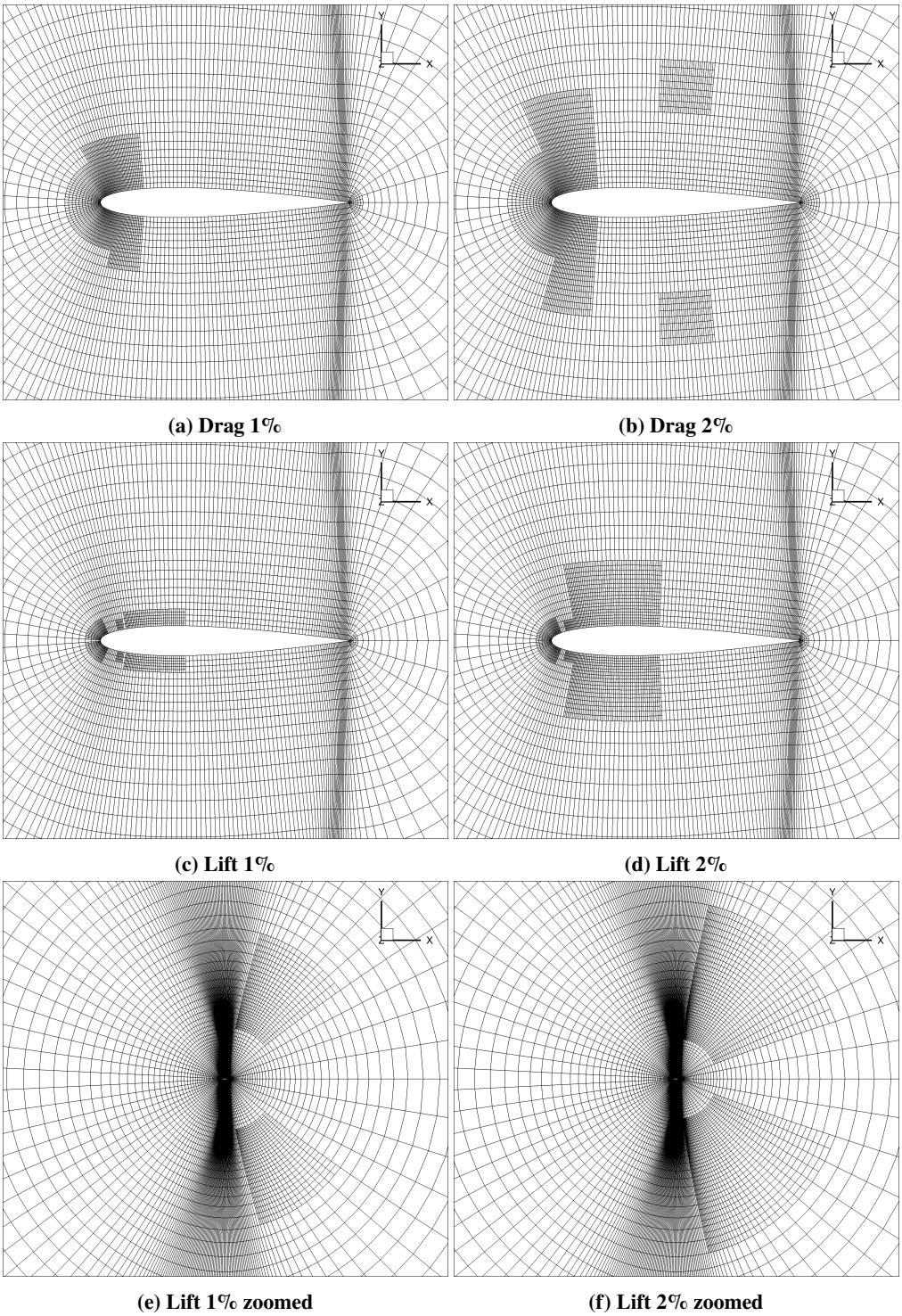


(a) x-momentum component of drag adjoint contours from -0.0005 to 0.0005

(b) x-momentum component of lift adjoint contours from -7 to 7

Fig. 10 x-momentum component of lift and drag adjoint on $L2$ mesh for NACA 0012 airfoil $M_\infty = 0.95$ $\alpha = 0°$.

The final adapted meshes on lift and drag with both 1% and 2% fixed fractions are shown below in Figure 11. Adapting on drag flagging by 1% of cells yields refinement around the leading and trailing edges. Increasing the fraction to 2% increases the size of the existing refinement blocks while also adding refinement around the lambda structure in the adjoint. Adapting based on lift yields refinement blocks down stream around the the oblique shock. This is due to large magnitudes in the coarse-space residual rather than the adjoint which is near zero in that region. Once the cells near the airfoil that affect lift and drag the most have been refined, cells in the far-field are flagged. This indicates that a lower adaptive fraction can be used. The lift and drag convergence plots are shown in Figure 12.

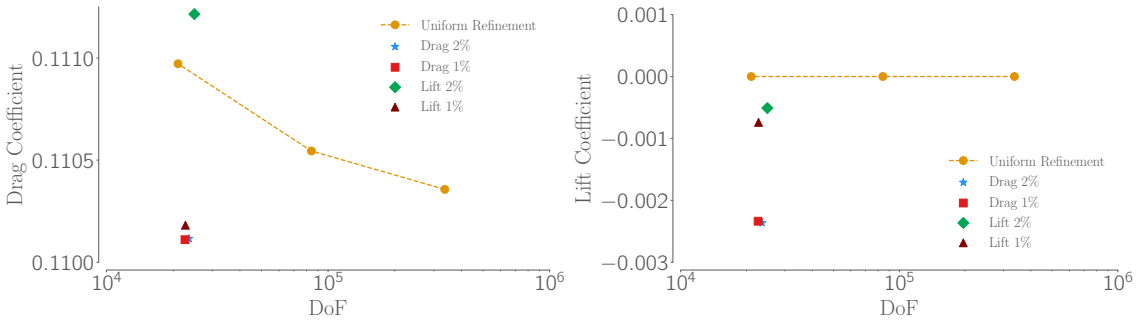


(a) Drag 1% **(b) Drag 2%**

(c) Lift 1% **(d) Lift 2%**

(e) Lift 1% zoomed **(f) Lift 2% zoomed**

Fig. 11 Mesh generated from adapting L2 mesh on lift and drag with 1% and 2% fixed fractions. $M_\infty = 0.95$ $\alpha = 2°$**.**



(a) Drag coefficient convergence. **(b) Lift coefficient convergence.**

Fig. 12 Output convergence for NACA 0012 Airfoil $M_\infty = 0.95$ $\alpha = 0°$.

Here we again see the drag prediction outperforming the $L1$ mesh in three of the four cases, and those even predict lower drag than the $L0$ mesh. The mesh adapting 2% of cells based on lift over-predicts drag and this could be due to the fact that there is more overlap between the refinement blocks, causing interpolation error. All cases have worse lift predictions, while the meshes that are adapted based on the lift perform better than the meshes adapting the drag error. The loss of accuracy in the lift makes sense, as the adaptation algorithm does not guarantee any symmetry between the upper and lower surfaces. The low lift predicted by the uniform meshes is 0 due to the symmetry of the mesh.

C. $M_\infty = 0.8$ $\alpha = 1.25°$ NACA 0012 Airfoil

The final test case we ran on the NACA 0012 airfoil is at transonic conditions with a Mach number $M_\infty = 0.8$ at $\alpha = 1.25°$. Mach contours are shown on the $L0$ mesh in Figure 13. There is a large strong shock on the upper surface of the airfoil and a weaker shock on the lower surface. Again, we adapt based on lift and drag on the cells with the highest

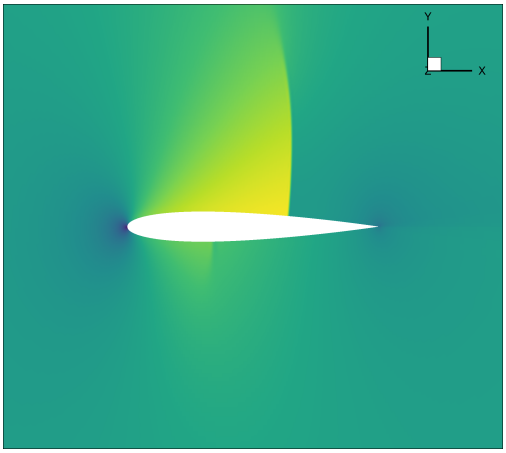
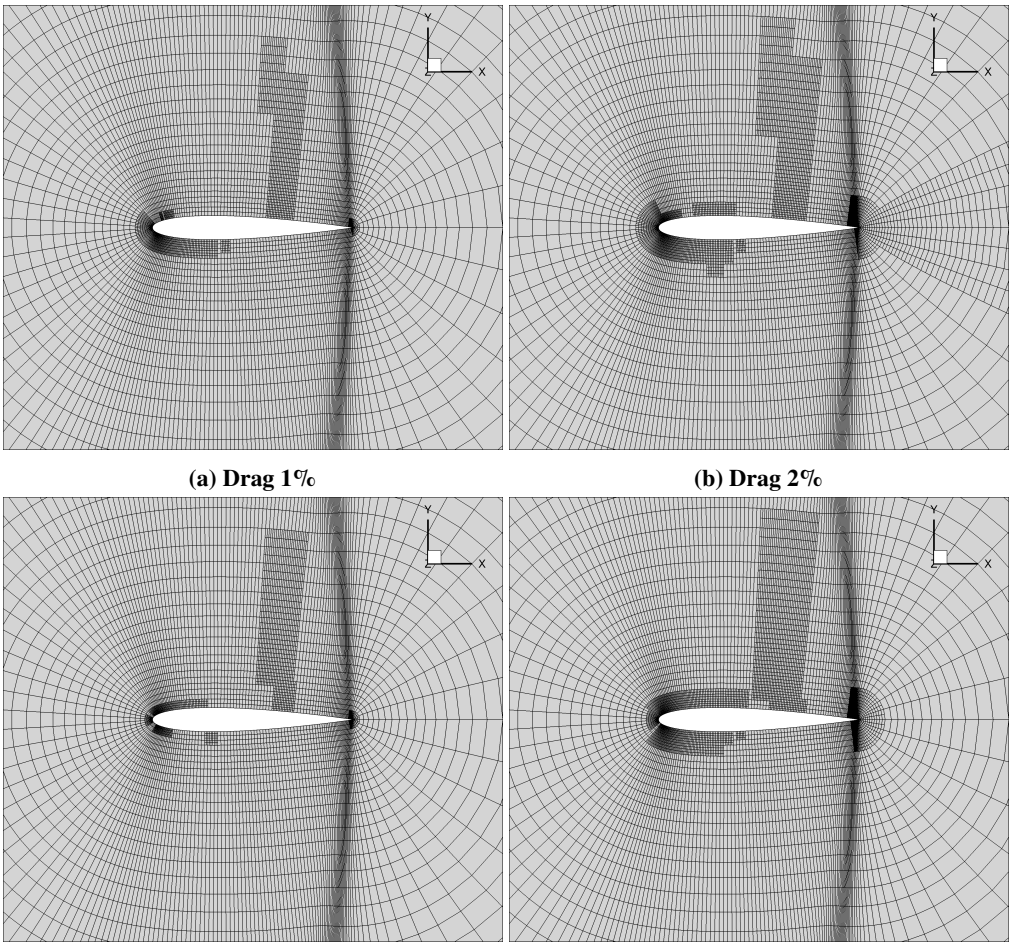


Fig. 13 Mach contours on $L0$ mesh. color range from 0 to 1.4.

1% and 2% of error for each case. The meshes adapted meshes for this test case are shown below in Figure 14. Here, we see that the meshes are refining the both the leading and trailing edges as well as around the shocks on the upper and lower surfaces.

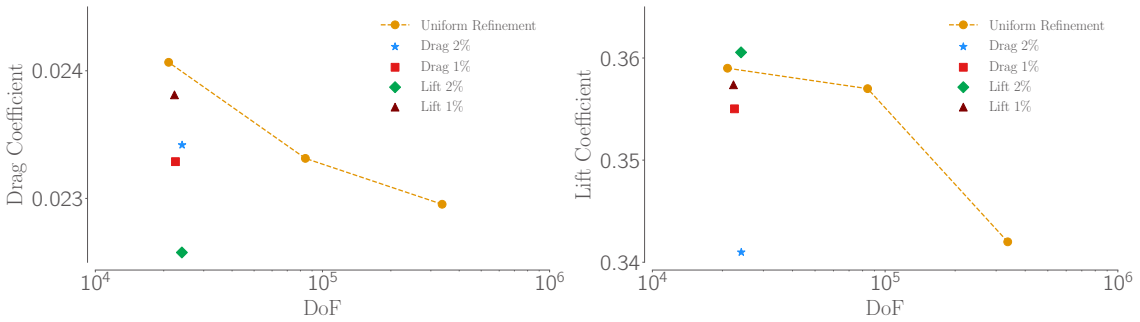
Lift and drag convergence plots are shown in Figure 15.



(a) Drag 1% **(b) Drag 2%**

(c) Lift 1% **(d) Lift 2%**

Fig. 14 Mesh generated from adapting L2 mesh on lift and drag with 1% and 2% fixed fractions. $M_\infty = 0.8$ $\alpha = 1.25°$**.**



(a) Drag coefficient convergence. **(b) Lift coefficient convergence.**

Fig. 15 convergence for NACA 0012 Airfoil $M_\infty = 0.8$ $\alpha = 1.25°$.

Here we see a more expected result where 2 of the test cases drag prediction are between $L2$ and $L1$ and one of the results is nearly identical to $L1$. The lift 2% case predicts drag lower than $L0$ by almost 4 drag counts and this could be due to dissipation being removed at the overset boundaries. Lift shows a similar behavior. Three of the adapted meshes are around the $L2$ and $L1$ range while the drag 2% case predicts lift closer to and slightly below the $L0$ mesh. These cases show that adapting on drag improves drag prediction better than lift prediction and adapting on lift predicts lift better than adapting on drag as expected. It also shows that in some cases with overset boundaries, it is possible to add too much refinement where the 1% case performs better than the 2% case. This could be due to the fact that larger overset boundaries are removing more entropy from the domain lowering the drag prediction.

VI. Conclusion

Structured meshes have many advantages over unstructured meshes for use in aerodynamic shape optimization due to their increased memory efficiency. However, structured meshes have many drawbacks due to the difficulties of meshing complex geometries. Overset meshes have been used as a way to combine overlapping structured meshes in an unstructured way to ease mesh generation difficulties. Implicit hole cutting can be used to fully automate generating the grid connectivity between overlapping overset meshes. Structured meshes still have draw backs when a mesh does not fully resolve important flow features. Due to the structured storage, nodes cannot be added locally and re-meshing of components or in some cases global re-meshing is required.

We presented an output-based mesh adaptation algorithm using overset meshes to provide local refinement to structured meshes. The discrete adjoint vector is used with a coarse-space residual to provide an adaptive indicator that is able to flag flow features that affect output error. A greedy algorithm was then developed to group cells flagged for refinement in a way that reduces refinement around cells that are not flagged as well as inter-grid communication. Numerical results show that adding overset blocks as refinement is able to improve drag predictions in a way that is non-intuitive by reducing drag more than just uniform refinement with only one adaptation iteration. This was shown to be a result of overset boundaries removing spurious entropy generated in the domain and therefore reducing drag. While this property is not ideal for the stability of inviscid solutions where negative entropy generation can lead to instabilities, spurious entropy generation is not the dominant source of drag or entropy in RANS simulations.

Acknowledgments

The first author would like to acknowledge the support received through various stages of the research by the Department of Defense (DoD) through the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program and the Rackham Graduate School at University of Michigan through the Rackham Merit Fellowship. The first author also received partial support from the University of Michigan Institute for Computational Discovery and Engineering Fellowship.

References

[1] Kenway, G. K. W., Secco, N., Martins, J. R. R. A., Mishra, A., and Duraisamy, K., "An Efficient Parallel Overset Method for Aerodynamic Shape Optimization," *Proceedings of the 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech Forum*, Grapevine, TX, 2017. doi:10.2514/6.2017-0357.

[2] Buning, P., and Pulliam, T., "Cartesian Off-Body Grid Adaption for Viscous Time-Accurate Flow Simulations," *20th AIAA Computational Fluid Dynamics Conference*, 2012. doi:https://doi.org/10.2514/6.2011-3693.

[3] Buning, P. G., and Pulliam, T. H., "Near-Body Grid Adaption for Overset Grids," *46th AIAA Fluid Dynamics Conference*, 2016. doi:https://doi.org/10.2514/6.2016-3326.

[4] Park, M. A., "Adjoint-Based, Three-Dimensional Error Prediction and Grid Adaptation," AIAA Paper 2002-3286, 2002.

[5] Fidkowski, K. J., and Darmofal, D. L., "Review of output-based error estimation and mesh adaptation in computational fluid dynamics," *AIAA Journal*, Vol. 49, No. 4, 2011, pp. 673–694. doi:10.2514/1.J050073.

[6] Venditti, D. A., and Darmofal, D. L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, No. 1, 2002, pp. 40–69. doi:https://doi.org/10.1006/jcph.2001.6967.

[7] Venditti, D. A., and Darmofal, D. L., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22–46. doi:https://doi.org/10.1016/S0021-9991(03)00074-3.

[8] Mader, C. A., Kenway, G. K. W., Yildirim, A., and Martins, J. R. R. A., "ADflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization," *Journal of Aerospace Information Systems*, Vol. 17, No. 9, 2020, pp. 508–527. doi:10.2514/1.I010796.

[9] Kenway, G. K. W., Mader, C. A., He, P., and Martins, J. R. R. A., "Effective Adjoint Approaches for Computational Fluid Dynamics," *Progress in Aerospace Sciences*, Vol. 110, 2019, p. 100542. doi:10.1016/j.paerosci.2019.05.002.

[10] Yildirim, A., Kenway, G. K. W., Mader, C. A., and Martins, J. R. R. A., "A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations," *Journal of Computational Physics*, Vol. 397, 2019, p. 108741. doi:10.1016/j.jcp.2019.06.018.

[11] Secco, N. R., Jasa, J. P., Kenway, G. K. W., and Martins, J. R. R. A., "Component-based Geometry Manipulation for Aerodynamic Shape Optimization with Overset Meshes," *AIAA Journal*, Vol. 56, No. 9, 2018, pp. 3667–3679. doi:10.2514/1.J056550.

[12] Secco, N., Kenway, G. K. W., He, P., Mader, C. A., and Martins, J. R. R. A., "Efficient Mesh Generation and Deformation for Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 59, No. 4, 2021, pp. 1151–1168. doi:10.2514/1.J059491.

[13] Luke, E., Collins, E., and Blades, E., "A Fast Mesh Deformation Method Using Explicit Interpolation," *Journal of Computational Physics*, Vol. 231, No. 2, 2012, pp. 586–601. doi:10.1016/j.jcp.2011.09.021.

[14] Becker, R., and Rannacher, R., "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, edited by A. Iserles, Cambridge University Press, 2001, pp. 1–102.

[15] Martins, J. R. R. A., and Ning, A., *Engineering Design Optimization*, Cambridge University Press, 2021. URL `https://mdobook.github.io`.

[16] Ding, K., Fidkowski, K. J., and Roe, P. L., "Acceleration Techniques for Adjoint-Based Error Estimation and Mesh Adaptation," Eighth International Conference on Computational Fluid Dynamics ICCFD8-0249, 2014.

[17] Oswatitsch, K., *Gas Dynamics*, Academic, New York, 1956.

[18] Benek, J. A., Steger, J. L., Dougherty, F. C., and Buning, P., "Chimera. A Grid-Embedding Technique," 1986.