University of Michigan XFlow Group
Correspondence: Steve Kast, *kastsm@umich.edu* PI: Krzysztof Fidkowski

C1.4 Laminar Boundary Layer on a Flat Plate

1. Code description

XFlow is a high-order discontinuous Galerkin (DG) finite element solver written in ANSI C, intended to be run on Linux-type platforms. Relevant supported equation sets include compressible Euler, Navier-Stokes, and RANS with the Spalart-Allmaras model. High-order is achieved compactly within elements using various high-order bases on triangles, tetrahedra, quadrilaterals, and hexahedra. Parallel runs are supported using domain partitioning and MPI communication. Visual post-processing is performed with an in-house plotter. Output-based adaptivity is available using discrete adjoints.

2. Case summary

The default implicit Newton solver was used for all runs in this case. The residual was converged to an absolute L_1 norm below 10^{-9} using a conservative state vector of $\mathcal{O}(1)$ freestream density, velocity, and pressure, and gas constant $R = 0.4$. Both uniform and adaptive refinement procedures were used to obtain a converged drag coefficient. Runs were performed on the *flux* supercomputing cluster at the University of Michigan. The number of cores ranged from 3 on the coarsest meshes to 80 on the finest meshes.

3. Meshes

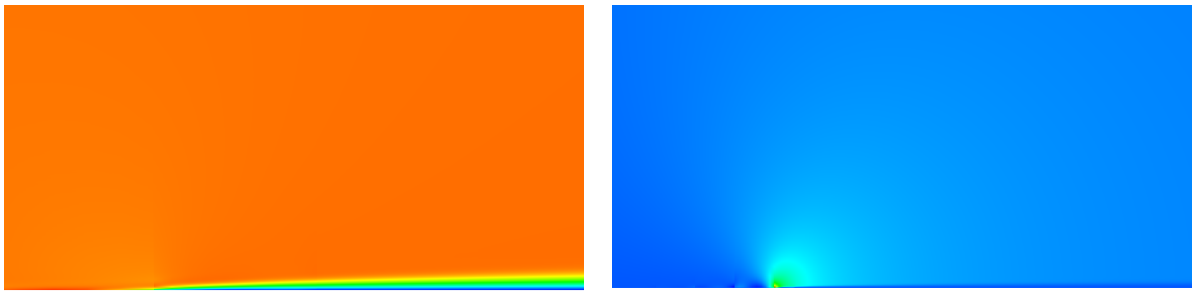
A series of triangular meshes was used for the uniform refinement sequence, ranging from 480 elements to 122880. For the adaptive runs, an initial mesh with 160 quadrilateral elements was used. The boundary distances are the same as those posted on the website – i.e. $L_H = 1.25$ and $L_V = 2$.

4. Results

The following figures and tables show the requested data for this case. Uniform refinement results are shown first, followed by adaptive runs.

Uniform Refinement

Representative solution contours are shown in Figure 1. A thin boundary layer develops near the plate, and the flow diverts slightly upward. Convergence plots showing both C_d and the error in C_d



(a) x-momentum (b) y-momentum

Figure 1: Momentum (ρu and ρv) contours near the plate.

are provided in Figures 2 and 3. To compute the error, a truth value of $C_d = 0.001312846$ was used, which is the final corrected value obtained from our *hp*-adaptive run.

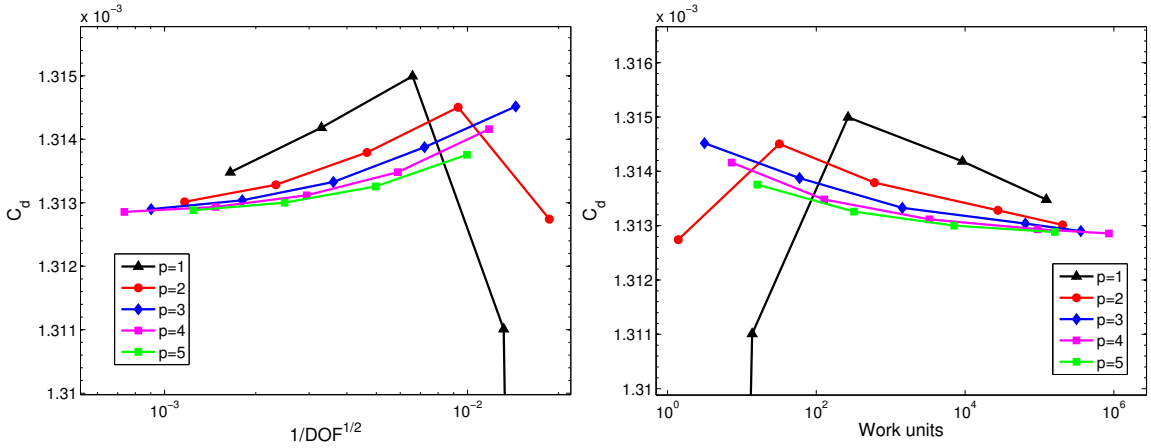


Figure 2: Drag coefficient convergence with uniform refinement.

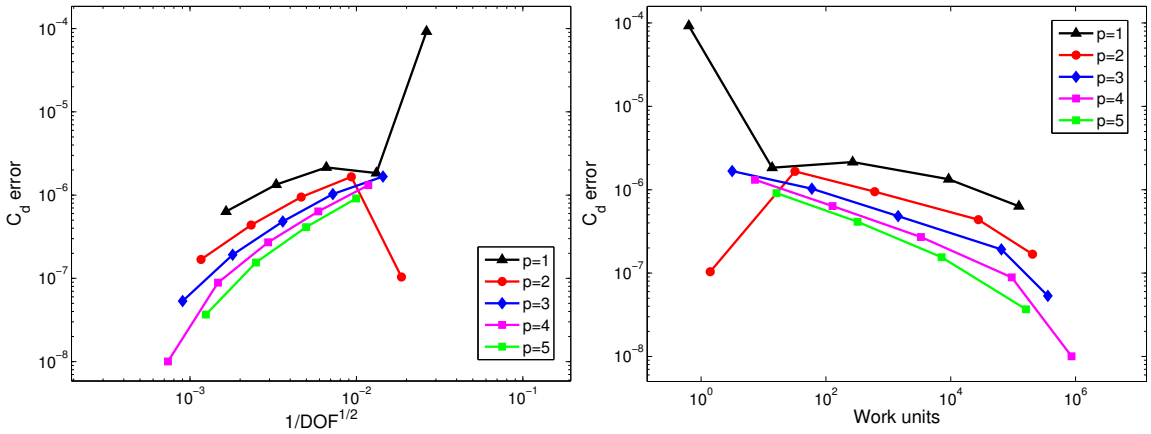


Figure 3: Convergence of drag coefficient error with uniform refinement.

	Ref=0	Ref=1	Ref=2	Ref=3	Ref=4
$p = 1$	1.2204e-3	1.3110e-3	1.3150e-3	1.3142e-3	1.3135e-3
Rate	-	5.6532	-0.22720	0.68434	1.0766
Work	6.3436e-01	1.3727e1	2.6763e2	9.2370e3	1.2348e5
$p = 2$	1.3127e-3	1.3145e-3	1.3138e-3	1.3133e-3	1.3130e-3
Rate	-	-3.9982	0.80885	1.1149	1.3746
Work	1.3965	3.1885e1	6.0511e2	2.7619e4	2.0434e5
$p = 3$	1.3145e-3	1.3139e-3	1.3133e-3	1.3130e-3	1.3129e-3
Rate	-	0.70030	1.0942	1.3265	1.8498
Work	3.1454	5.9489e1	1.4343e3	6.4568e4	3.5955e5
$p = 4$	1.3142e-3	1.3135e-3	1.3131e-3	1.3129e-3	1.3129e-3
Rate	-	1.0463	1.2332	1.6100	3.1427
Work	7.2952	1.2762e2	3.3204e3	9.4695e4	8.5962e5
$p = 5$	1.3138e-3	1.3133e-3	1.3130e-3	1.3129e-3	-
Rate	-	1.1433	1.4119	2.0760	-
Work	1.6128e1	3.2149e2	7.1260e3	1.6062e5	-

Table 1: Drag coefficients from the uniform refinement runs, along with rates and work units.

Table 1 gives the drag coefficient values, work units, and convergence rates for all uniform refinement runs. In general, the convergence rates are suboptimal, though they are still increasing as of the final adaptation. This is expected, due to the singularity at the leading edge.

Adaptive Refinement

Next, we perform adaptive mesh refinement, with several indicators used for comparison. These indicators consist of (i) the entropy variables (which serve as an adjoint to an entropy-flux output [1]), (ii) the unweighted residuals, (iii) the drag adjoint with p-adaptation, (iv) the drag adjoint with hp-adaptation [2], and (v) uniform p- and h-refinements. Figures 4 and 5 show the convergence for each of these methods as a function of both mesh size and work units.

In terms of **mesh size**, the most efficient methods in order are:

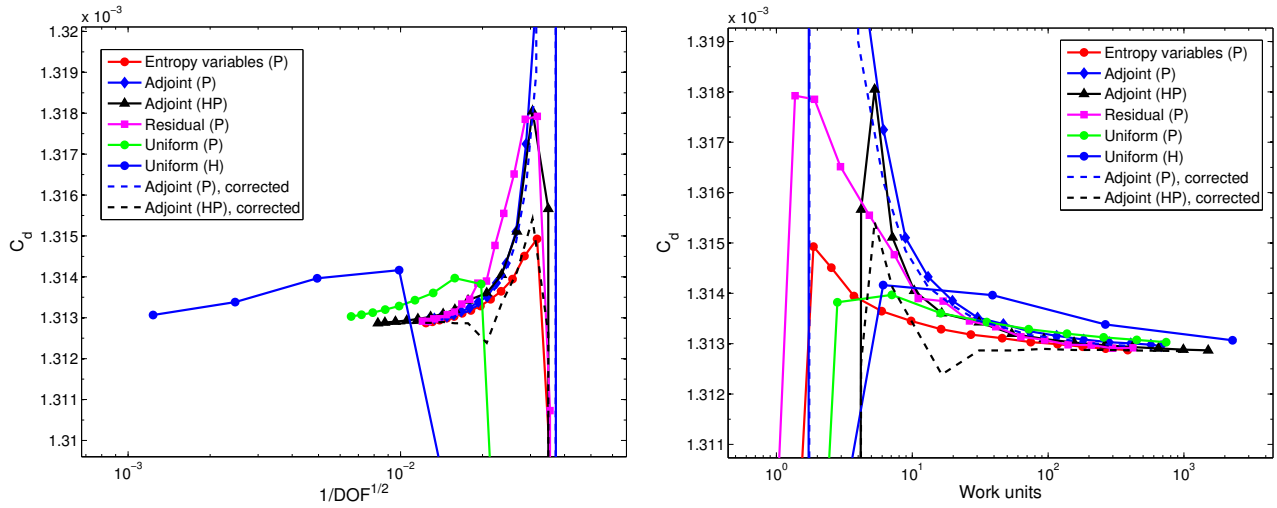
1. adjoint-based hp (corrected with error estimate)
2. entropy variables p
3. adjoint-based p
4. residual p
5. uniform p
6. uniform h

In terms of **work units**, the most efficient methods are:

1. adjoint-based hp (corrected with error estimate)
2. entropy variables p
3. residual p
4. adjoint-based p
5. uniform p
6. uniform h

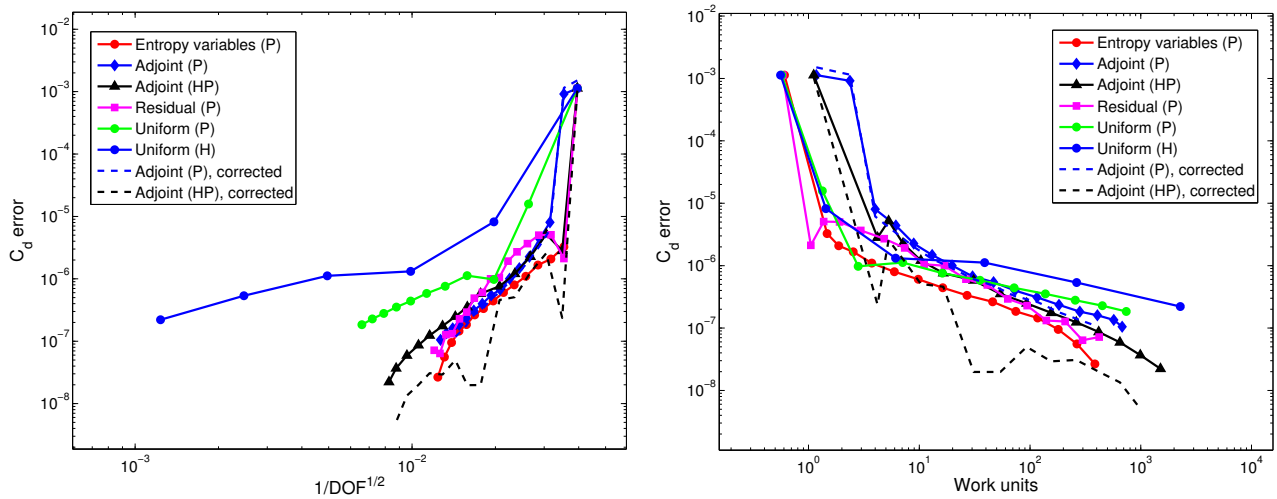
The best methods overall are therefore the adjoint-based hp-adaptation and the entropy variable adaptation. It is interesting that although the entropy variables do not serve as an adjoint to the actual drag output, they are still one of the most efficient adaptive methods. This is likely due to the connection between the entropy flux and drag through the Oswatitsch formula (see e.g. [1]).

Finally, adapted meshes for each method are shown in Figure 6. The stagnation point at the plate leading edge is heavily refined by all methods, and appears to be the primary contributor to the error in the drag coefficient.



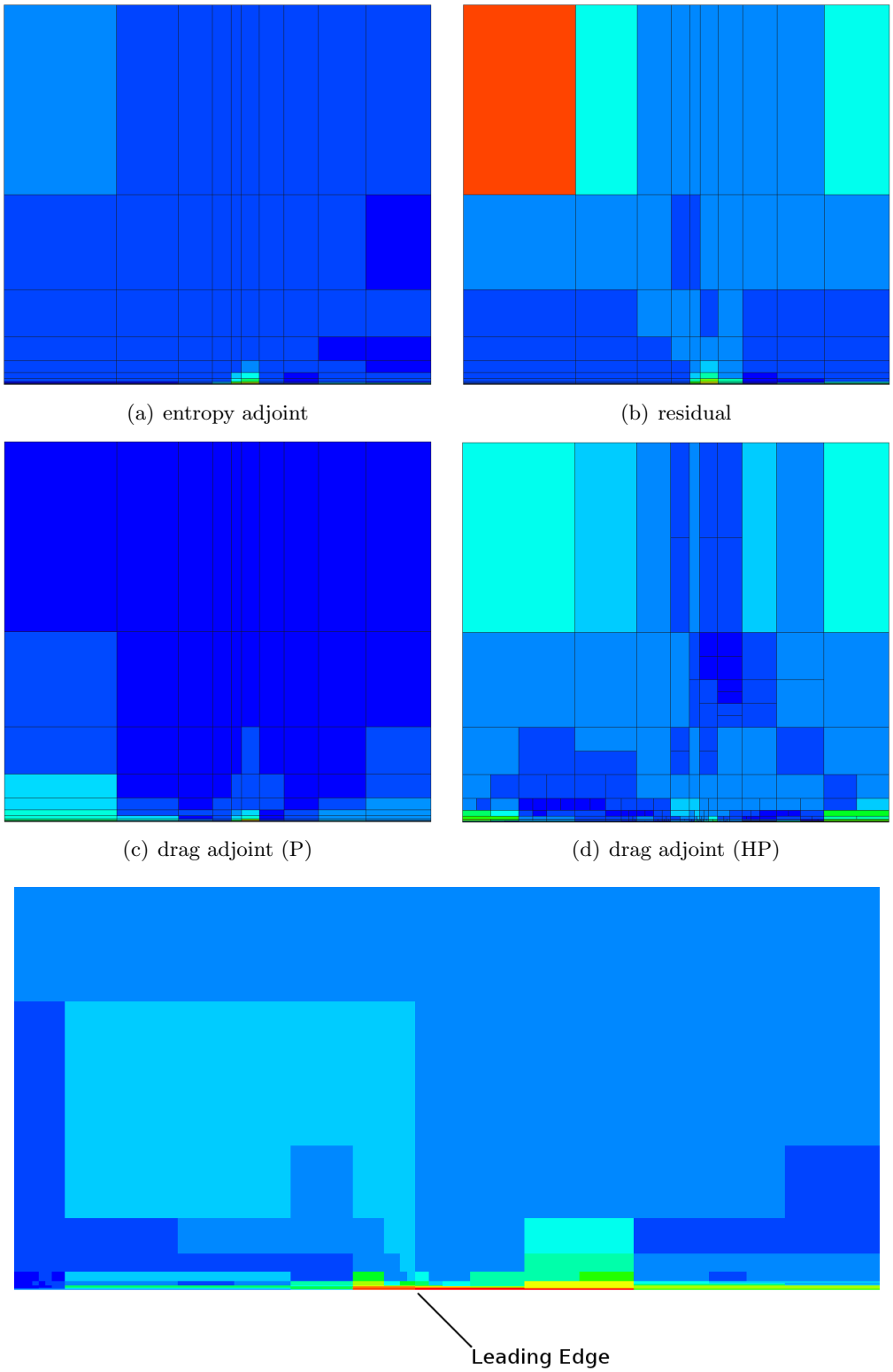
(a) (b)

Figure 4: Convergence of drag coefficient with adaptive refinement.



(a) (b)

Figure 5: Convergence of drag coefficient error with adaptive refinement.



(a) entropy adjoint

(b) residual

(c) drag adjoint (P)

(d) drag adjoint (HP)

(e) drag adjoint (HP) zoom

Figure 6: Adapted meshes for the various methods. The orders range from $p = 1$ (dark blue) to $p = 16$ (red). While difficult to see in the figures, there is significant adaptation in the small elements near the plate leading edge.

References

[1] K. J. Fidkowski, M. A. Ceze, P. L. Roe, Entropy-based drag error estimation and mesh adaptation in two dimensions, AIAA Journal of Aircraft 49 (2012) 1485–1496.

[2] M. A. Ceze, K. J. Fidkowski, Anisotropic hp-adaptation framework for functional prediction, AIAA Journal 51 (2012) 492–509.