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

C1.1 Internal inviscid flow over a smooth bump

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 case consists of subsonic ($M = 0.5$) inviscid flow over a smooth bump. The entropy error is used as a measure of overall solution accuracy. We provide both uniform refinement and adaptive results based on several metrics.

The default implicit Newton solver was used for all runs. The residual was converged to an absolute L_1 norm below 10^{-10} using a conservative state vector of $\mathcal{O}(1)$ freestream density, velocity, and pressure, and gas constant $R = 1.0$. On each mesh, runs were first solved at $p = 0$, and the order was then incremented sequentially up to $p = 5$. For a given p, the work units reported represent the cumulative work to reach that p.

The runs were performed on the *flux* supercomputing cluster at the University of Michigan. One Taubench unit on this machine corresponds to 9.08 seconds of compute time. The number of cores ranged from 1 on the coarsest meshes to 12 on the finest meshes.

3. Meshes

The meshes used for uniform refinement were generated in-house, and consist of quadrilateral elements with Q=4 geometry representation (see Figure 1 for an example). The number of elements ranged from 256 to 65536. For the adaptive runs, the second mesh in the sequence (with 1024 elements) was used.

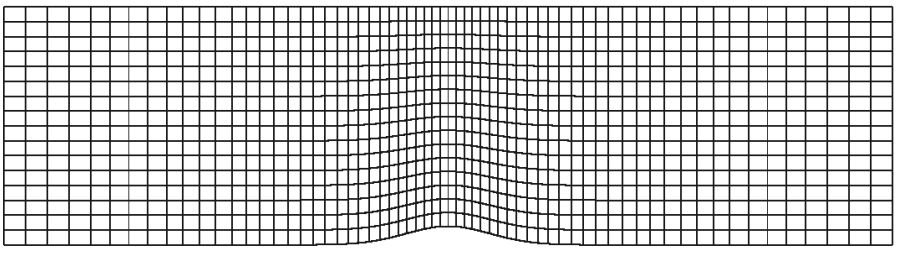


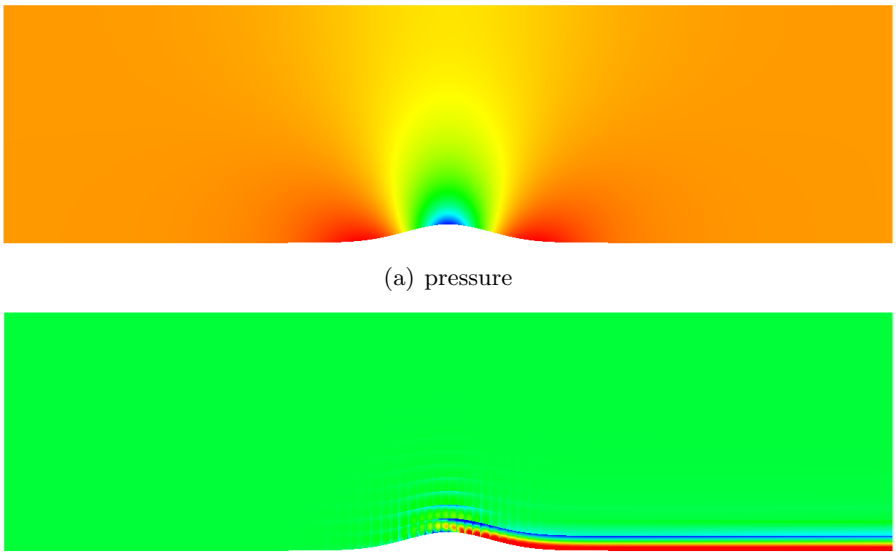
Figure 1: Sample mesh used in the uniform refinement study. This mesh is the second in a series of 5 refinements.

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 2. We see that most of the spurious entropy is generated near the bump itself, and then advected downstream. Convergence results for various p- and h-refinements are provided in Figure 3. Finally, rates and work units are given in Table 1. Expected convergence rates are achieved for all orders.



(a) pressure

(b) entropy

Figure 2: Pressure and entropy contours. The entropy is spurious, and is advected downstream after being generated near the bump.

Adaptive Refinement

Next, we perform adaptive mesh refinement, with several indicators used for comparison. These indicators consist of (i) the entropy itself, (ii) the entropy variables (which serve as an adjoint to an entropy-flux output [1]), (iii) the unweighted residuals, (iv) the adjoint for the entropy error, and (v) uniform p- and h-refinements. Figure 4 shows the convergence for each of these methods in terms of both mesh size and work units.

In terms of mesh size, we see that the entropy variables, entropy error adjoint, and residual perform the best, with the adjoint converging the fastest by a slim margin. Uniform p-refinement also does well (given the smooth nature of the problem), while pure entropy adaptation and uniform h-refinement perform the worst. In terms of work units, the residual adaptation is the fastest overall, followed closely by uniform p-refinement.

Adapted meshes for each method are shown in Figure 5. The adjoint, residual, and entropy variables adapt similar regions, focusing refinement near the bump itself. The pure entropy adaptation, however, is distracted by the entropy errors advecting downstream, and therefore never adapts their actual *source*. This is expected – we included this adaptation primarily to contrast it with the behavior

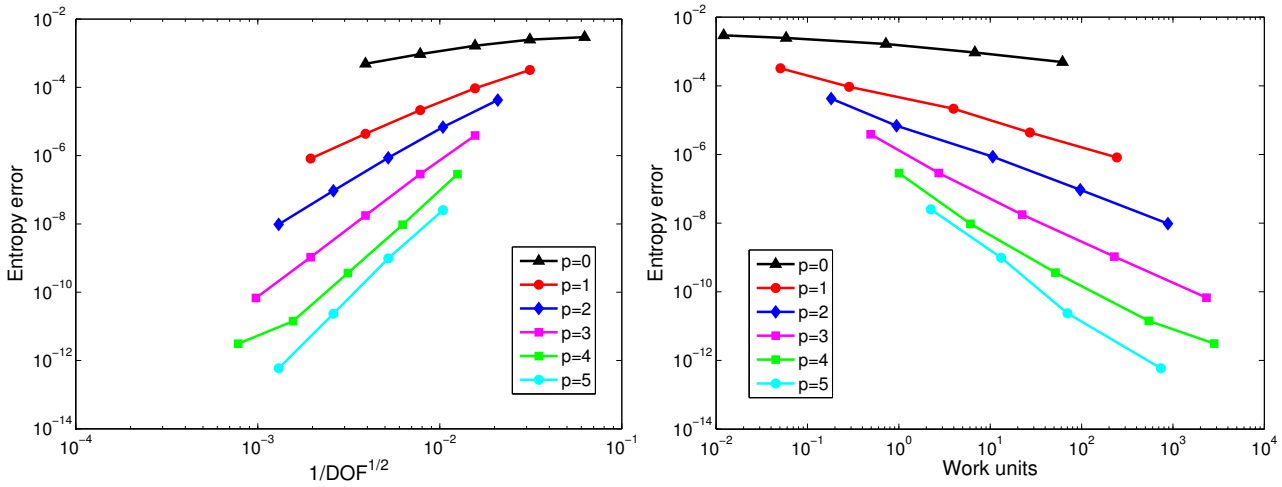
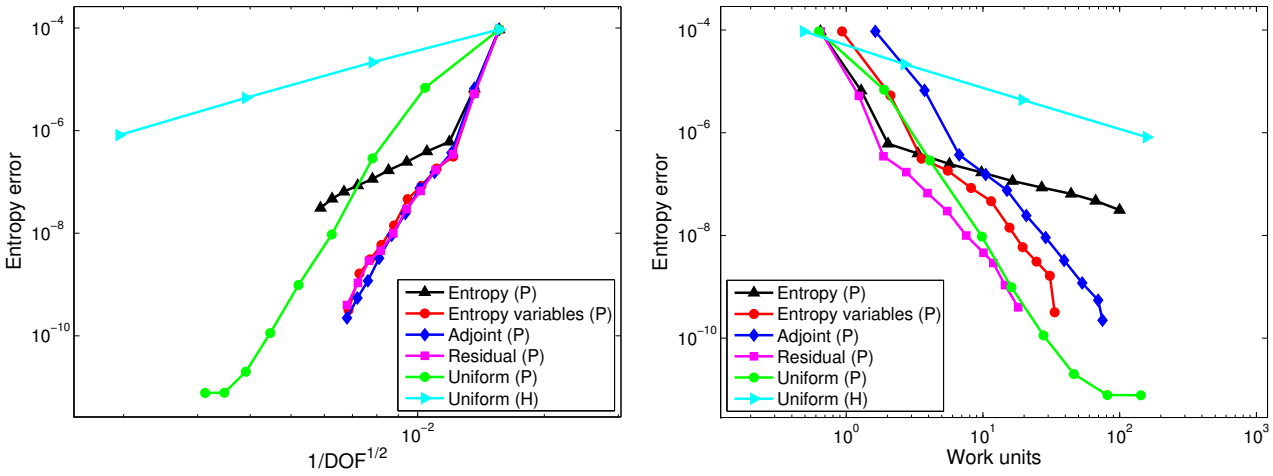


Figure 3: Convergence of entropy error with h and p refinement.

Table 1: Uniform refinement data.

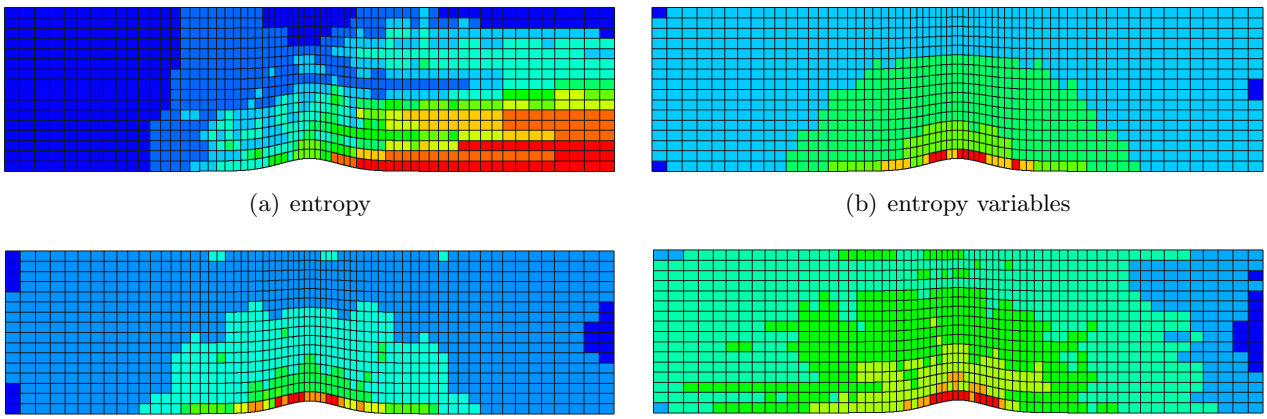
	Ref=0	Ref=1	Ref=2	Ref=3	Ref=4
$p = 0$	2.9477e-3	2.4878e-3	1.6554e-3	9.3641e-4	4.9085e-4
Rate	-	0.2447	0.58771	0.82196	0.93185
Work	1.2115e-2	5.8370e-2	7.2247e-1	6.8018	6.2009e+1
$p = 1$	3.2343e-4	9.3860e-5	2.1559e-5	4.3591e-6	8.1994e-7
Rate	-	1.7849	2.1222	2.3062	2.4104
Work	5.0661e-2	2.8634e-1	3.9648	2.7119e+1	2.4325e+2
$p = 2$	4.2463e-5	6.8588e-6	8.6102e-7	9.3884e-8	9.6710e-9
Rate	-	2.6302	2.9939	3.1971	3.2792
Work	1.8172e-1	9.4493e-1	1.0665e+1	9.6775e+1	8.8139e+2
$p = 3$	3.8812e-6	2.8843e-7	1.7698e-8	1.0603e-9	6.7582e-11
Rate	-	3.7502	4.0265	4.0611	3.9717
Work	4.9229e-1	2.7401	2.2357e+1	2.2833e+2	2.3280e+3
$p = 4$	2.8712e-7	9.4848e-9	3.6310e-10	1.4167e-11	3.1086e-12
Rate	-	4.9199	4.7072	4.6797	2.1882
Work	1.0044	6.0815	5.1670e+1	5.4395e+2	2.8246e+3
$p = 5$	2.5280e-8	9.8232e-10	2.3518e-11	5.9162e-13	-
Rate	-	4.6857	5.3844	5.3130	-
Work	2.2368	1.3198e+1	7.0088e+1	7.3763e+2	-

of the adjoint and the entropy variables, both of which target the source of errors rather than their effects.



(a) (b)

Figure 4: Entropy error convergence in terms of both (a) mesh size and (b) work units for various adaptive methods. Note that an "H" or "P" in the legend indicates the type of adaptation performed. For H adaptation, an order of $p = 1$ was used.



(a) entropy (b) entropy variables

(c) residual (d) adjoint

Figure 5: Final meshes for the various adaptive methods (blue regions are low order, red are high order). The entropy variables, residual, and adjoint refine similar regions near the bump, while pure entropy adaptation targets the entropy advected downstream.

References

[1] K. J. Fidkowski, P. L. Roe, An entropy adjoint approach to mesh refinement, SIAM Journal on Scientific Computing 32 (3) (2010) 1261–1287.