C 1.1 Smooth Bump

Marshall C. Galbraith*, Paul D. Orkwis†
University of Cincinnati, Cincinnati, OH, 45221

and

John A. Benek‡
U.S. Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, 45433

I. Code Description

The solver relies a standard Discontinuous Galerkin discretization to achieve high-orders of accuracy. The spatial fluxes are approximated using the Roe approximate Riemann solver.[1] All integrals are evaluated analytically during a pre-processing stage using a symbolic manipulation package. The fluxes are computed from the conservative variables using a modal representation of the polynomials rather than a traditional nodal method. A more detailed description of the discretization and solver can be found in Refs. 2 and 3. The code is currently capable of up to 4^{th}-order polynomial approximations of the solution, which yields a 5^{th}-order accurate scheme. There is no inherent limitation that prevents polynomial approximations of higher order.

A linearized form of Euler equations is solved using an implicit Newtons method. The implicit system of equations of the Newtons method is solved using a flexible version of the GMRES iterative matrix solution algorithm, FGMRES.[4] A block ILU1[4] preconditioner is used for the FGMRES algorithm. The FGMRES algorithm was limited to 100 inner iterations. The FGMRES solver was converged to a residual below 10^{-14} on each Newton iteration. All calculations are in double precision.

Parallel capabilities are forthcoming.

II. Case Summary

The residual for the mean coefficient of the continuity equation, i.e. the density residual, from the first iteration was multiplied with 1×10^{-10} to yield the tolerance for a converged solution. The solution was initialized to a uniform flow, and the solution was converge to a solution to steady state without a temporal term in the Euler equations. This is equivalent to $CFL = \infty$. A total inflow boundary condition was imposed on the inlet, and free-stream pressure was imposed on the outlet. Slip wall boundary conditions were imposed on the upper and lower boundaries.

A summary of the exucation times and the average execution time of TauBench is shown in Table 1. The code was executed on an a machine with two 6 cores Intel i7 Xeon X5660 2.8 GHz processors. Only one core was used for all calculations.

*PhD Candidate, School of Aerospace Systems, AIAA Student Member.
†Professor of Aerospace Engineering & Engineering Mechanics, School of Aerospace Systems, AIAA Associate Fellow.
‡Director, Computational Sciences Center, Air Vehicles Directorate, Air Force Research Laboratory, AIAA Fellow.

Run	Execution Time (s)
1	7.815
2	7.674
3	7.698
4	7.962
5	7.669
6	7.712
7	7.691
8	7.953
9	7.676
10	7.668
Avg:	7.752

Table 1: Taubench Results

Timing of 100 right hand side (RHS) evaluations with 250,000 degrees of freedom (DOF) are summarized in Table 2. The RHS was evaluated on a uniform rectangular mesh. The solution was initialized to a free-stream flow with slip walls on the lower and upper surfaces. As the order of the polynomial approximation increased, the cell size of the mesh was reduced in order to maintain constant DOF. The 100 RHS evaluations were achieved by marching with a 4^{th}-order Runge-Kutta scheme 25 iterations.

Polynomial Order	Mesh Size	DOF	Work
0	500x500	250,000	6.24
1	250x250	250,000	2.70
2	167x167	251,001	2.58
3	125x125	250,000	3.32
4	100x100	250,000	4.92

Table 2: RHS Timing

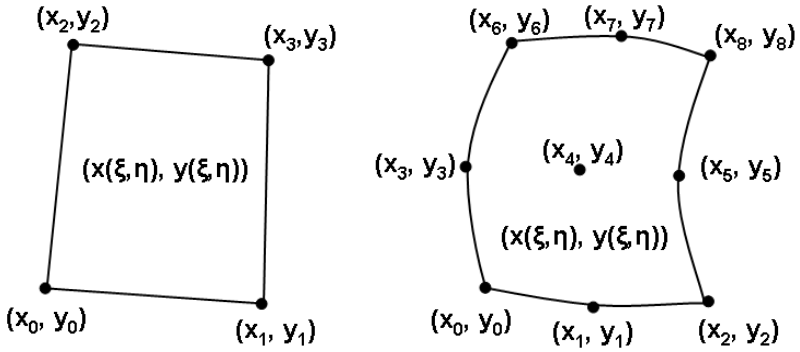
III. Meshes

The DG solver is formulated for structured meshes. These meshes can be generated using any traditional mesh generator. After reading the mesh coordinates, the solver generates cell local polynomial representations, $(x(\xi,\eta), y(\xi,\eta))$, of the cell coordinates. The geometric polynomial mapping of the cell coordinates is formulated as a sum involving the same test functions, ψ, as used in the DG discretization of the governing equations. Hence,

$$\begin{aligned} x(\xi,\eta) &= \sum_{i}^{N_g}\sum_{j}^{N_g} x_{ij}\psi_{ij}(\xi,\eta) \\ y(\xi,\eta) &= \sum_{i}^{N_g}\sum_{j}^{N_g} y_{ij}\psi_{ij}(\xi,\eta) \end{aligned} \tag{1}$$

where x_{ij} and y_{ij} are the coefficients of the expansion, and $\psi_{ij}(\xi,\eta) = P_i(\xi)P_j(\eta)$, where $P_i(\xi)$ are the Legendre polynomials. The coefficients are found by equating the expansion with the associated cell nodal

values. The process is repeated for the y coordinate to obtain the final polynomial mapping of the cell. Additional points are required to establish the polynomial representation as shown for a quadratic cell in Fig. 1b. This implies that the grid must consist of $N_g n+1$ points along a coordinate line to generate n cells with a geometric mapping of order N_g along that line.



(a) Linear cell (b) Quadratic cell

Figure 1: Nodal and modal representation of cells.

The meshes were generated with a uniform grid spacing in both the x and y coordinates with the algorithm of Table 3.

```
node_imax = cell_imax*Ng + 1
node_jmax = cell_jmax*Ng + 1

y1 = 0.8

DO j = 0 to j = node_jmax-1
        DO i = 0 to i = node_imax-1
                x(i,j) = -1.5 + (3*i)/(node_imax-1)

                y0 = 0.0625*exp(-25*x(i,j)*x(i,j))

                y(i,j) = y0 + (y1-y0)*j/(node_jmax-1)
        END DO
END DO
```

Table 3: Mesh Generation Algorithm

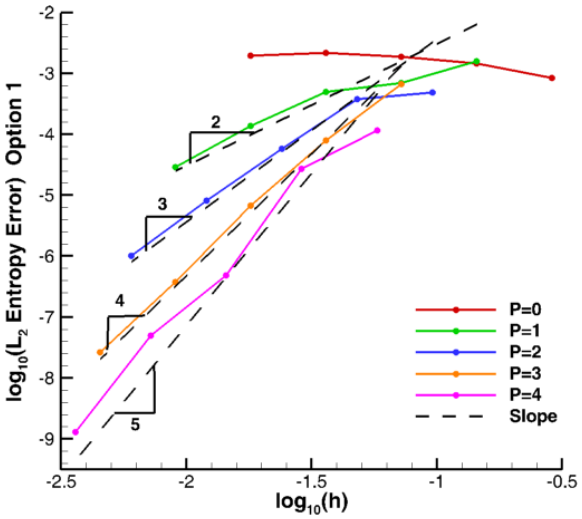
where N_g is the order of the geometric polynomial mapping, cell_imax is the number of cells in the x-coordinate direction, and cell_jmax is the number of cells in the y-coordinate direction. The mesh sizes used for the calculations are summarized in Table 4. The uniform grids spacing of the mesh generation algorithm automatically achieves a uniform grid refinement.

6x2	12x4	24x8	48x16	96x32

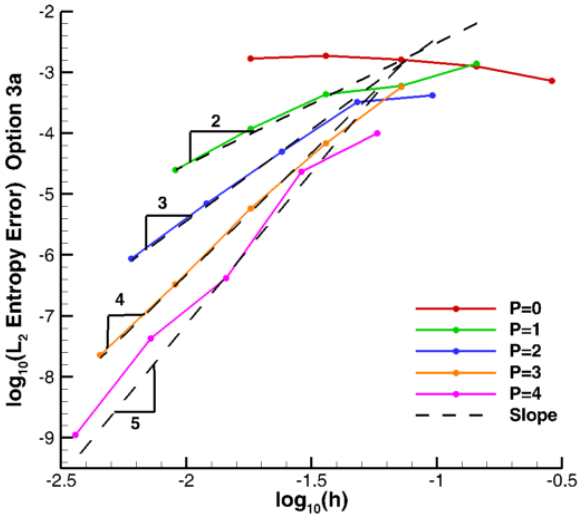
Table 4: Mesh Sizes Used for the Calculations

IV. Results

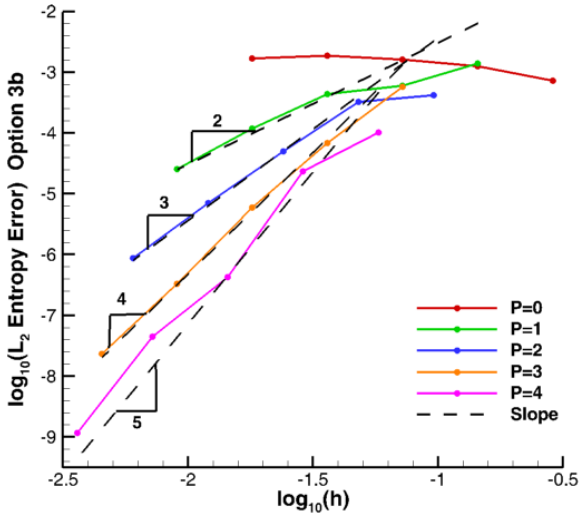
The convergence rate of three types of the L_2-error options are shown in Fig. 2. Except for the polynomial approximation $P = 0$, the convergence rate of the solution achieves the expect $O\left(h^{P+1}\right)$. L_2-errors vs. Work are shown in Fig. 3. With few exceptions, a higher degree polynomial achieves lower L_2-errors with less work. The convergence history of the Newton solver for the different meshes is shown in Fig. 4. For each mesh, the total number of iterations reach a converged solution remains relatively unchanged with increased orders of P. The number of iterations to reach convergence does tend to increase with the increase in mesh size.



(a) Option 1

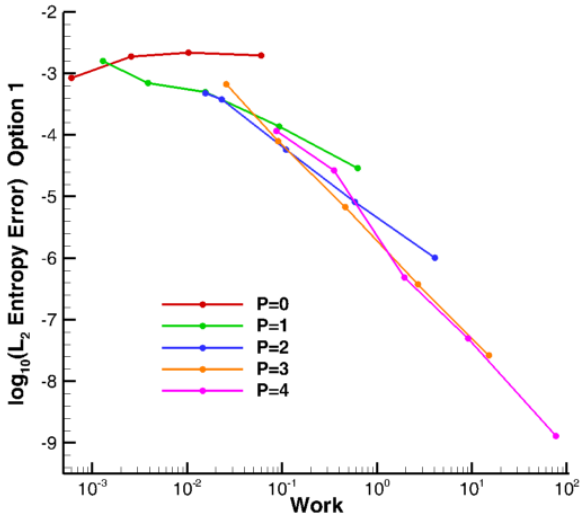


(b) Option 3a

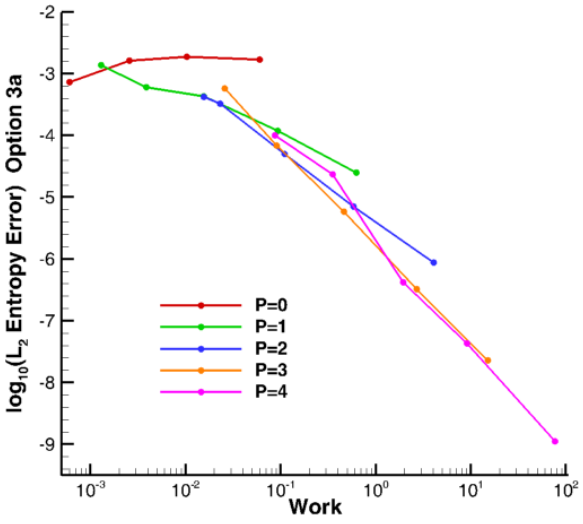


(c) Option 3b

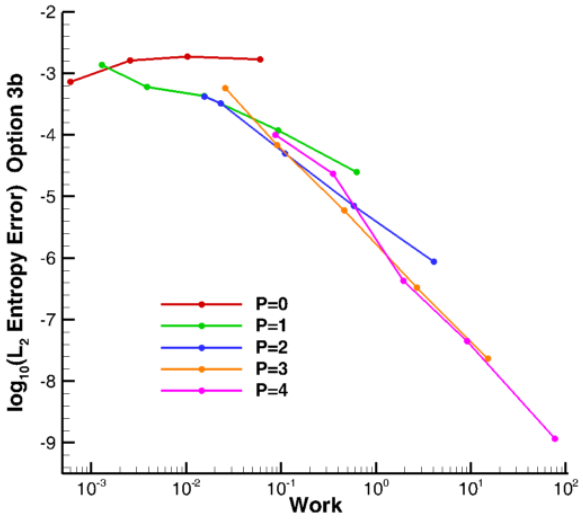
Figure 2: H-Convergence for $N_g = 4$



(a) Option 1

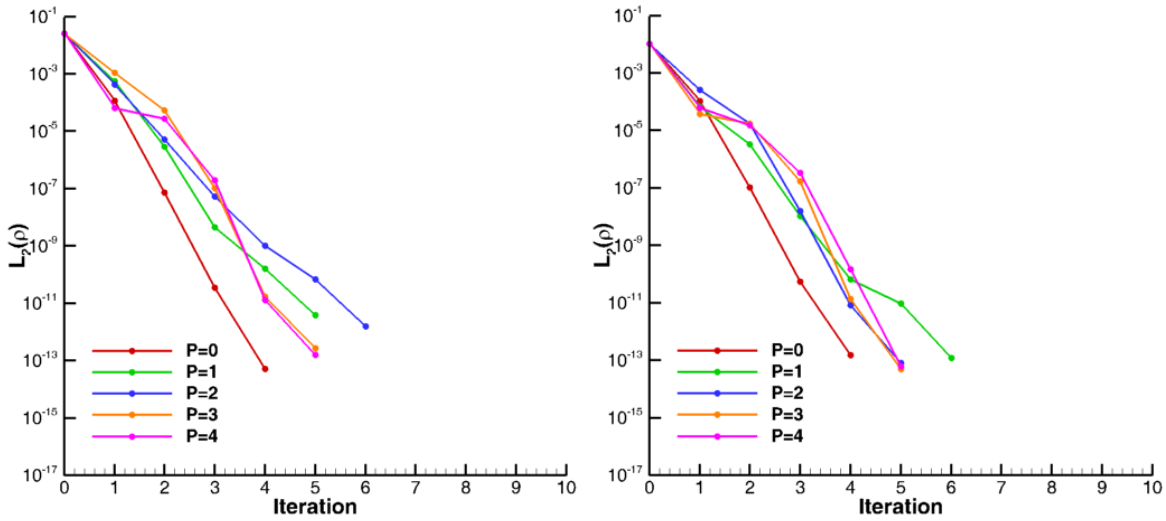


(b) Option 3a



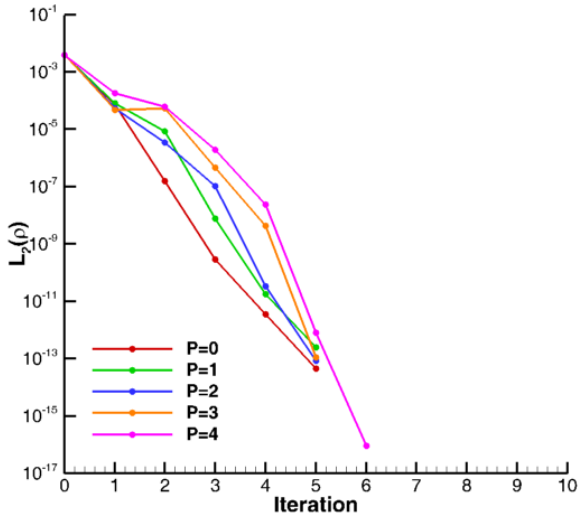
(c) Option 3b

Figure 3: Work for $N_g = 4$

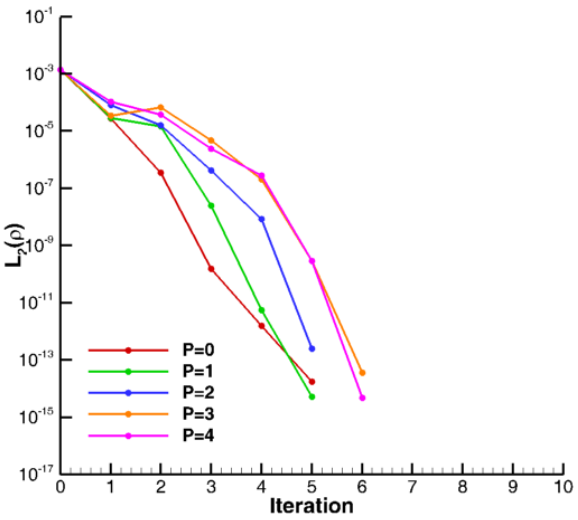


(a) 6x2

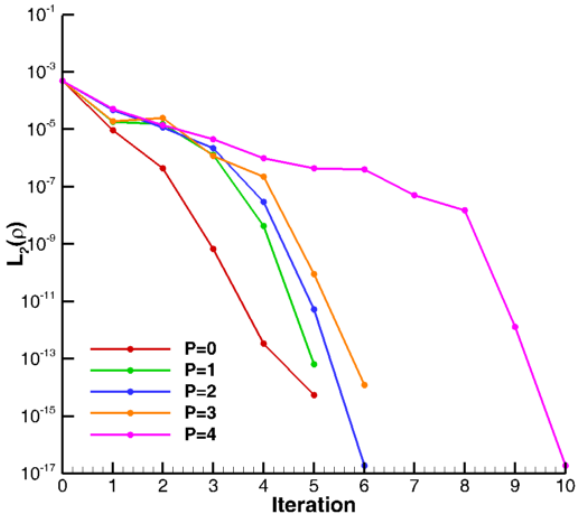
(b) 12x4



(c) 24x8



(d) 48x16



(e) 96x32

Figure 4: Newton Iteration Convergence History for $N_g = 4$

References

[1]Vatsa, V. N. and Wedan, J. L. T. B. W., "Navier-Stokes computations of prolate spheroids at angle of attack," AIAA-Paper 1987-2627, 1987.

[2]Galbraith, M. C., Orkwis, P. D., and Benek, J. A., "Automated Quadrature-free Discontinuous Galerkin Method Applied to Viscous Flows," AIAA-Paper 2011-493, 2011.

[3]Galbraith, M. C., Orkwis, P. D., and Benek, J. A., "Extending the Discontinuous Galerkin Scheme to the Chimera Overset Method," AIAA-Paper 2011-3409, 2011.

[4]Saad, Y., *Iterative methods for sparse linear systems*, SIAM, 2nd ed., 2000.