C1.6 Vortex Transport by Uniform Flow

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 prescribed initial condition was imposed via least-squares projection onto the space spanned by the DG basis at each order. Time stepping was performed using a fourth order diagonally-implicit Runge-Kutta time stepping scheme. At each stage, the nonlinear system was solved using implicit Newton preconditioned with a line-Jacobi smoother.

At each stage of the time stepping scheme, the residual was converged to an absolute L_1 norm below 10^{-12} using a conservative state vector. The freestream quantities provided in the problem description are in SI units, but we found that setting up the problem in these units made adequate iterative convergence of the momentum equations difficult to achieve. This is because the code monitors the L_1 norm of the entire state vector for convergence, and due to the large values of energy and limits of machine precision, driving this norm below $\sim 5 \times 10^{-7}$ was not feasible for all meshes. Therefore, the input conditions were normalized using $R_{\text{gas}} = 1, p_\infty = 1, T_\infty = 1$. This means that our nondimensional velocities are related to the physical velocities via the factor $\sqrt{(287.15J/kg.K)(300K)}$, and hence this factor was used to multiply the computed L_2 errors for consistency.

Runs were performed on the *nyx* supercomputing cluster at the University of Michigan. The number of cores ranged from 64 on the coarsest meshes to 192 on the finest meshes. On one core of the nyx machine, one TauBench unit is equivalent to 16.5 seconds of compute time.

3. Meshes

Triangular meshes were constructed in-house using a Matlab script. This script created a uniform lattice over the square domain and then created right triangles by bisecting each square along one of the diagonals. Random perturbations to the mesh (for the requested case) were added according to the specified $\delta = 0.15h$. We note that the perturbed mesh sequence is not nested in that the perturbations were imposed independently on each mesh in the sequence. Sample meshes are shown in Figure 1.

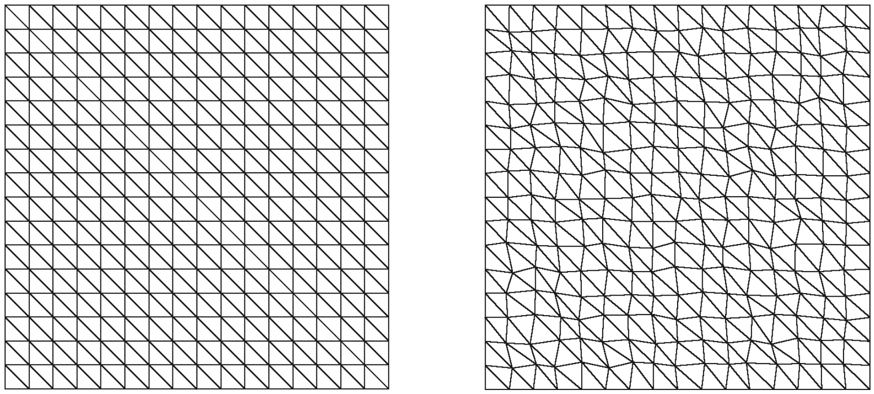
4. Results

The figures and tables below present the requested results. DIRK4 was used for the temporal discretization: the number of time steps was set to 1600 for the coarsest mesh (8×8), and doubled for every successively-refined mesh. This may have been somewhat excessive for the coarse orders but it avoided tuning at different orders.

<u>Slow vortex: $M_\infty = .05, \beta = .02, R = 0.005$</u>

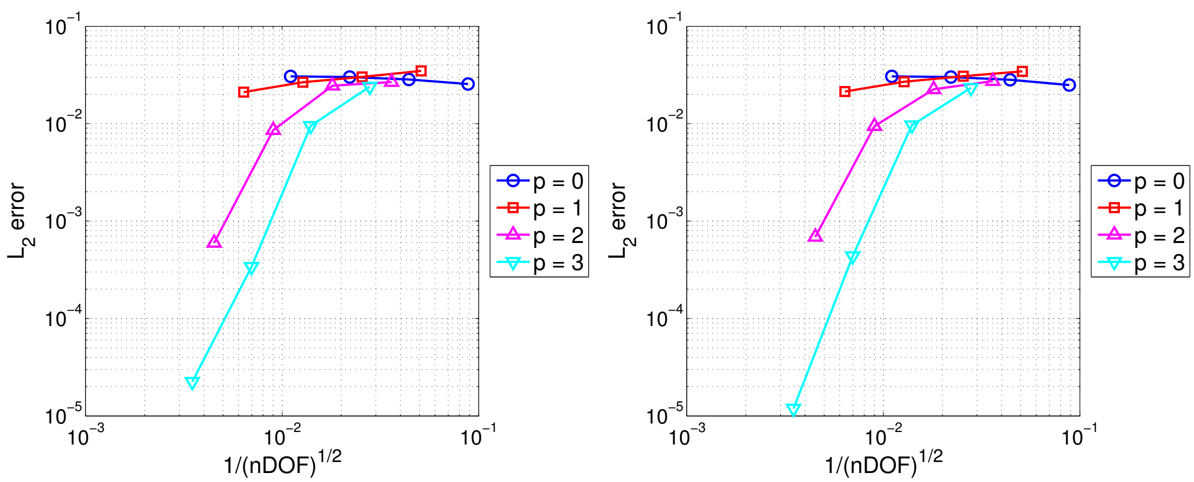
Figure 2 shows convergence of the velocity L_2 error of the velocity field with mesh refinement.. Figure 3 shows the same results versus work units. Tables 1 and 2 show the same data in tabular form.

<u>Fast vortex: $M_\infty = .5, \beta = .2, R = 0.005$</u>



(a) Unperturbed mesh (b) Perturbed mesh

Figure 1: Sample unperturbed and perturbed meshes generated for this case.



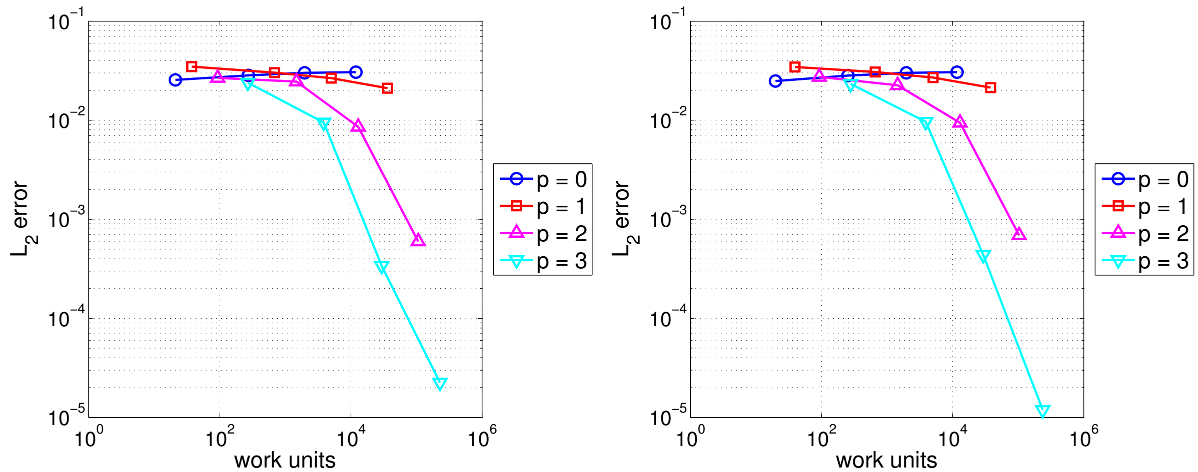
(a) Uniform mesh (b) Perturbed mesh

Figure 2: $M = 0.05, \beta = .02, R = 0.005$: L_2 error of the velocity field, plotted versus a function of the degrees of freedom.

Table 1: $M = 0.05, \beta = .02, R = 0.005$: L_2 error of the velocity field.

nelem	p = 0	p = 1	p = 2	p = 3
128	2.5513e-02	3.4804e-02	2.6765e-02	2.3969e-02
rate	-	-	-	-
512	2.8395e-02	3.0200e-02	2.4535e-02	9.4855e-03
rate	-0.15	0.20	0.13	1.34
2048	3.0112e-02	2.6634e-02	8.5876e-03	3.3916e-04
rate	-0.08	0.18	1.51	4.81
8192	3.0576e-02	2.1069e-02	5.9813e-04	2.2432e-05
rate	-0.02	0.34	3.84	3.92

Figure 4 shows convergence of the velocity L_2 error of the velocity field with mesh refinement.. Figure 5 shows the same results versus work units. Tables 3 and 4 show the same data in tabular



(a) Uniform mesh (b) Perturbed mesh

Figure 3: $M = 0.05, \beta = .02, R = 0.005$: L_2 error of the velocity field, plotted versus work units.

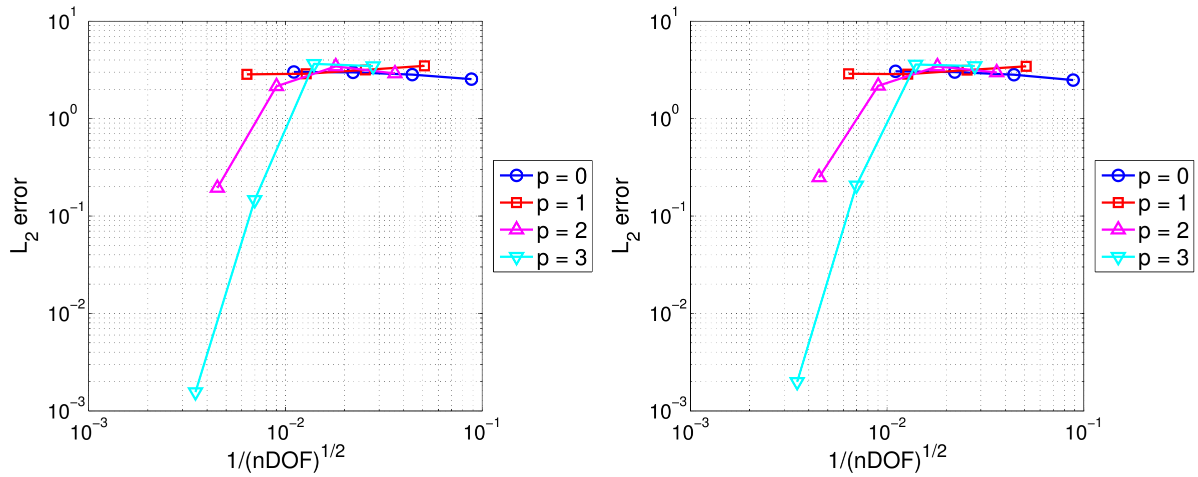
Table 2: $M = 0.05, \beta = .02, R = 0.005$: L_2 error of the velocity field, perturbed mesh.

nelem	p = 0	p = 1	p = 2	p = 3
128	2.4900e-02	3.4458e-02	2.7366e-02	2.3302e-02
rate	-	-	-	-
512	2.8220e-02	3.0710e-02	2.2548e-02	9.6376e-03
rate	-0.18	0.17	0.28	1.27
2048	3.0090e-02	2.6947e-02	9.4376e-03	4.3670e-04
rate	-0.09	0.19	1.26	4.46
8192	3.0589e-02	2.1354e-02	6.8883e-04	1.1911e-05
rate	-0.02	0.34	3.78	5.20

form.

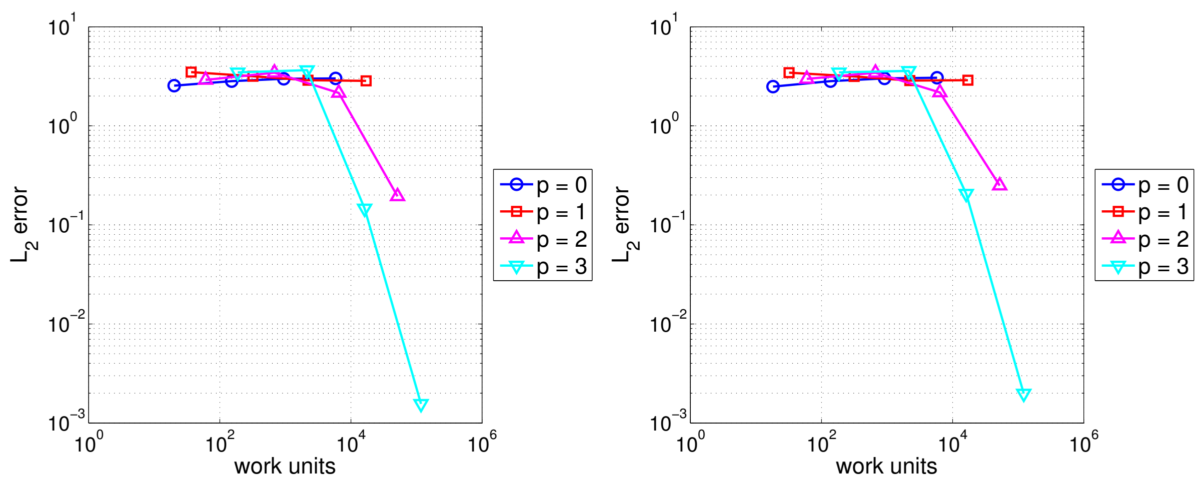
Table 3: $M = 0.5, \beta = .2, R = 0.005$: L_2 error of the velocity field.

nelem	p = 0	p = 1	p = 2	p = 3
128	2.5422e+00	3.4847e+00	2.9073e+00	3.4619e+00
rate	-	-	-	-
512	2.8269e+00	3.1647e+00	3.4386e+00	3.6491e+00
rate	-0.15	0.14	-0.24	-0.08
2048	2.9853e+00	2.8899e+00	2.1465e+00	1.4650e-01
rate	-0.08	0.13	0.68	4.64
8192	3.0101e+00	2.8459e+00	1.9414e-01	1.5597e-03
rate	-0.01	0.02	3.47	6.55



(a) Uniform mesh (b) Perturbed mesh

Figure 4: $M = 0.5, \beta = .2, R = 0.005$: L_2 error of the velocity field, plotted versus a function of the degrees of freedom.



(a) Uniform mesh (b) Perturbed mesh

Figure 5: $M = 0.5, \beta = .2, R = 0.005$: L_2 error of the velocity field, plotted versus work units.

Table 4: $M = 0.5, \beta = .2, R = 0.005$: L_2 error of the velocity field, perturbed mesh.

nelem	p = 0	p = 1	p = 2	p = 3
128	2.4887e+00	3.4423e+00	2.9796e+00	3.4647e+00
rate	-	-	-	-
512	2.8218e+00	3.1485e+00	3.4319e+00	3.5880e+00
rate	-0.18	0.13	-0.20	-0.05
2048	3.0080e+00	2.8632e+00	2.1699e+00	2.0651e-01
rate	-0.09	0.14	0.66	4.12
8192	3.0571e+00	2.8899e+00	2.4941e-01	1.9834e-03
rate	-0.02	-0.01	3.12	6.70