

ENC-2022-0540

EVALUATION OF A PARALLEL HIGH-ORDER CFD SOLVER DEVELOPED USING CHAPEL

Fábio Malacco Moreira

Instituto Tecnológico de Aeronáutica, 12228-900 São José dos Campos, SP, Brazil
fabiom91@gmail.com

João Luiz F. Azevedo

Instituto de Aeronáutica e Espaço, 12228-904 São José dos Campos, SP, Brazil
joaluiz.azevedo@gmail.com

Abstract. *The Flux Reconstruction (FR) method is a recent approach to compact high-order spatial discretizations that unifies schemes such as the Discontinuous Galerkin (DG), Spectral Volume (SFV), and Spectral Difference (SD) methods into a single mathematical framework. It also allows for a wide range of new scheme variants to be developed inside this framework and it is generally of simpler computational implementation than its predecessors. The Chapel programming language is also a recent development that integrates parallel programming directives and controls natively into the language. It uses a Partitioned Global Address Space (PGAS) approach in which the data communication in a distributed memory cluster is hidden from the programmer, while still allowing fine control over data locality. The objective of this effort is to build a competitively performant, shared-memory parallel, 2-D Euler flow solver in Chapel using the FR method. While 2nd-order Finite Volume (FV) solvers are common and sufficient for many applications, especially steady-state flows, high-order methods are particularly well suited for problems for which spatial resolution is critical, such as Large Eddy Simulations (LES) and vortex dominated flows. The Chapel programming language offers an opportunity to simplify software development, reducing entry barriers for new researchers, increasing productivity, and, therefore, reducing time spent on computational optimization. Such a programming environment should also allow for easier scaling of the resulting code to a distributed memory solver in the future. The solver developed will be validated through comparisons of flow solutions to some standard CFD test cases, such as the Ringleb flow and airfoil calculations. These comparisons will take into consideration both solution accuracy and computational cost. The work demonstrates both the capability of the FR method in accurately resolving critical 1-D and 2-D benchmark problems and of the Chapel programming language as a viable alternative to the traditional MPI approach. The present effort can be seen as a proof of concept to allow for further development of more complex, 3-D solvers in the future.*

Keywords: *CFD, Parallel Programming, Unstructured High-Order Methods, Chapel, Flux Reconstruction*

1. INTRODUCTION

The Computational Fluid Dynamics (CFD) field is constantly evolving to enable more complex and detailed simulations, such as Large Eddy Simulations (LES). These simulations are fundamental for some applications such as combustion, acoustics and other highly turbulent flows and require high flow resolution. Achieving this level of flow resolution in simulations sometimes is possible just with considerable mesh refinement but in several cases more sophisticated, high-order methods are required to adequately capture the physics of the situation. Either way both of these alternatives result in increased computational demand and require applications that can take advantage of increased computational resources.

High performance, parallel software development in the scientific computing field traditionally relied on a combination of a well established procedural language, such as Fortran, C or C++, and a specialized external library that provides specific parallel programming features. The two most popular parallel programming libraries are MPI, which provides functions for network communication between several instances of a program, and OpenMP which modifies specific loops or code sections to split them into several tasks on a single system. To achieve ideal performance several production codes need to use a combination of both of these libraries. This combination of tools created a complicated development environment and require significant expertise for a developer to contribute to a given project.

While this barrier can be overcome, even if at significant cost, by organizations capable of building large, specialized and

long term team to build and maintain application it affect academic research particularly heavily since students may arrive with varying levels of programming experience and are expected to make significant contribution in a relatively short time span. The Chapel programming language provides a set of features that make it a promising alternative, especially when developer productivity is paramount. Primarily Chapel has native support for parallel programming dispensing MPI and OpenMP and providing high level abstractions for parallelism without restricting access to low level features if required. Secondly Chapel was designed from scratch targeted at high performance computing, incorporating features from Python, C, C++, Fortran, Java to provide a convenient environment for developing such applications.

The primary objective of the authors' work is to build and evaluate a modern and efficient CFD application capable of scaling from PC to cluster seamlessly based on these new parallel programming paradigms. As a mid-point in this project this work is aimed at evaluating a 2D CFD solver capable of accurately solving smooth inviscid flows and scaling through multiple cores in a shared memory system.. Tests that evaluate the accuracy of the numerical discretization scheme are performed along with basic aeronautically relevant flow simulations and code scalability tests.

2. NUMERICAL METHODS

2.1 Spatial Discretization

The Flux Reconstruction (FR) method is a spatial discretization approach first presented in Huynh (2007) for 1D conservation laws with extensions to quadrilateral and hexahedral elements using tensor products. In Huynh (2009), the FR method is further extended for diffusion problems enabling the discretization of the Navier-Stokes equations. Lastly, several extensions to other cell geometries were presented (Huynh, 2011; Wang and Gao, 2009 e Castonguay, 2012). In this work only the simplified formulation of the FR method to a 1D conservation law will be presented, although the applications of interest here are all 2-D. For a more detailed description of the method in higher dimensions, one should refer to the papers previously referenced.

Consider the 1-D conservation law

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = 0 \quad (1)$$

applied to a finite spatial domain Ω . The domain of the problem can be partitioned into a finite number of non-overlapping cells Ω_n . Then a mapping function can be devised to transform each cell from its original geometry in the physical domain to a standardized element in a computational domain. In the case of 1D meshes all cells can be converted to a unit segment $I = [-0.5, 0.5]$ by the function $x = x_n(\xi) = a_n * \xi + b_n$ where a_n and b_n are cell dependent parameters. An inverse transformation that maps the computational element to the physical domain can also be easily derived. Adequate transformations for more complex geometries can be found in texts on finite-element methods.

The solution of Eq. (1) can be approximated in each cell by a degree p polynomial. This polynomial is defined by its value in a set of $N_{sp} = p + 1$ nodes that will be called solution points (SPs). This approximation of the solution is smooth within each cell but can be discontinuous across cell interfaces. Currently although the code supports several SP distributions, all simulations are performed using Legendre-Gauss nodes as SPs. By using the same set of nodes ξ_i in the computational domain for every mesh element we obtain a set of universal interpolation coefficients for every cell that can be defined by

$$\ell_n(\xi) = \prod_{\substack{i=1 \\ i \neq n}}^{N_{sp}} \left(\frac{\xi - \xi_i}{\xi_n - \xi_i} \right). \quad (2)$$

These polynomial functions can be used to interpolate the solution inside each cell through a linear combination of the Lagrange basis weighted by the the nodal values of the solution

$$u(\xi) = \sum_{i=1}^{N_{sp}} (u_i \ell_i(\xi)). \quad (3)$$

Similarly to how this universal interpolation basis is calculated a basis for the derivative of the of this polynomial can be easily generated by derivating the Lagrange polynomial

$$\frac{du(\xi)}{d\xi} = \sum_{i=1}^{N_{sp}} \left(u_i \frac{d\ell_i(\xi)}{d\xi} \right) = \sum_{i=1}^{N_{sp}} (u_i \ell'_i(\xi)). \quad (4)$$

Which can be calculated applying the product rule to the expression in Eq. (2), taking the form of

$$\ell'_n(\xi) = \sum_{\substack{i=1 \\ i \neq n}}^{N_{sp}} \left[\left(\frac{1}{\xi_n - \xi_i} \right) \prod_{\substack{j=1 \\ j \neq i \\ j \neq n}}^{N_{sp}} \left(\frac{\xi - \xi_j}{\xi_n - \xi_j} \right) \right]. \quad (5)$$

With this basis, any function defined at these nodes can be interpolated or derived.

We turn now to evaluating the flux function in each cell and its derivative. With the values of the conserved variables at the SPs, we can calculate the value of the flux function in each node. Then using the Lagrange basis derived previously, these nodal values can be used to interpolate a flux polynomial with Eq. (3) and its derivative with Eq. (4). However, this interpolated flux function is based only on the internal state of the cell and may be discontinuous in the interfaces with other cells. Thus using only this discontinuous flux $f_d(\xi)$ to calculate the residual would lead to a violation of conservation and an incorrect solution. In order to fix this discontinuity, a flux correction $f_c(\xi)$ can be used to generate a new, continuous, flux function $F(\xi)$ as in

$$F(\xi) = f_d(\xi) + f_c(\xi). \quad (6)$$

This flux correction accounts for both left and right boundaries in the case of a 1D cell

$$f_c(\xi) = f_{c_{LB}}(\xi) + f_{c_{RB}}(\xi). \quad (7)$$

To calculate the corrections required to the discontinuous flux at the boundary, a common flux value must be defined for each interface. This common flux can be calculated using any Riemann solver and will be called f_{LB}^* and f_{RB}^* . Equation (3) can be used to obtain the left (u_L) and right (u_R) states of an interface which will be used by the Riemann solver $f^*(u_L, u_R)$. In this work, all simulations are performed using the Roe solver (Roe, 1981) with Harten's entropy fix (Harten, 1983). Previous experience of the authors with high-order methods suggests the impact of more refined Riemann solvers is diminishing, but further tests should be performed to verify this.

With the fluxes at the cell interfaces defined we can calculate the correction needed to the discontinuous flux

$$\begin{aligned} f_{LB}^* &= F(\xi_{LB}) \\ f_{LB}^* &= f_d(\xi_{LB}) + f_{c_{LB}}(\xi_{LB}) \\ f_{c_{LB}}(\xi_{LB}) &= f_{LB}^* - f_d(\xi_{LB}). \end{aligned} \quad (8)$$

This difference between the discontinuous flux and the numerical flux at the cell interface will be called the flux jump, J_{LB} , which can be used to scale a smooth correction function $g_L(\xi)$ restricted by the conditions

$$\begin{aligned} g_L(\xi_{LB}) &= 1, \\ g_L(\xi_{RB}) &= 0 \end{aligned} \quad (9)$$

and that approaches the zero function in the rest of the domain as to alter the cell flux as little as required to achieve continuity at the interface

$$\begin{aligned} f_{c_{LB}}(\xi) &= g_L(\xi) (f_{LB}^* - f_d(\xi_{LB})) \\ f_{c_{LB}}(\xi) &= g_L(\xi) J_{LB}. \end{aligned} \quad (10)$$

A similar procedure can be performed for the right boundary to obtain the continuous flux $F(\xi)$

$$F(\xi) = f_d(\xi) + g_L(\xi) J_{LB} + g_R(\xi) J_{RB}. \quad (11)$$

2.2 Temporal Discretization

Currently only explicit time stepping schemes of the Runge-Kutta family are implemented in the code. The schemes implemented include a 2nd-order arbitrary stage number SSP-RK scheme, a couple of 3rd-order accurate 4- and 5-stage SSP-RK schemes and a 4th-order, 5-stage SSP-RK, all of which were developed in Spiteri and Ruuth (2003). The numerical experiments in this work were all run using the 2nd-order scheme with 3 stages in order to maximize the CFL limit and shorten simulation time.

3. THE CHAPEL LANGUAGE

The Chapel programming language was developed with the objective of increasing programmer productivity in the development of parallel application, and specifically targeted at scientific and numerical computing. It was initially developed by Cray as part of the DARPA High Productivity Computing Systems program (HPCS) in the early 2000's and is currently developed under an open-source license mainly by Hewlett Packard Enterprise. It takes inspiration on several popular languages such as Python (iterators and other high-level conveniences), Fortran (array syntax), C (IO), C++ (object orientation) combining these concepts into a high-level environment for developing parallel application while allowing developers to access lower level abstractions when desired, or necessary, and therefore generating scalable and performant applications.

Chapel's parallelism model is based on a partitioned global address space (PGAS) paradigm, where there is a single global namespace for variables on every distributed memory node, which is called a locale in Chapel. When a Chapel application is executed, a single instance is started and initiates itself on all available locales. From thereon specific commands in the program are used to designate tasks that can be executed in parallel and how the parallelism should be handled. For example, a 'forall' command specifies that the iterations of a loop can be executed in parallel and the program may spread the execution of each iteration through several tasks, similarly to an OpenMP directive. A 'coforall' command, on the other hand, causes a separate task to be spawned for every iteration of the loop. As a last example, 'begin' and 'cobegin' statements can be used to spawn one or several, respectively, new tasks to execute a function or a block of code independently of any loop or iterative process. When using these directives, a new task is spawned to do the work and the main task resumes execution at the end of the code block as soon as the new task is spawned.

Array operations and data parallelism can be elegantly defined by a new concept introduced in Chapel of the array domain. Every array must be defined with an associated domain which may be constant or change though the program execution. The domain is the set of indices belonging to the array. It can be defined by a range starting and ending at any integer index and with any integer stride or it can be just a set of numbers individually added. The domain is used to manage array allocation, deallocation and reallocation, removing the need to manually manage memory. A domain may also have a distribution strategy which defines how it should be partitioned/spread across several separate memory partitions, adding a layer of abstraction over distributed memory that replaces typical MPI communication calls.

Besides the main parallelism features, Chapel provides support for generic functions and object orientation though record and class type variables. Record type variables use value semantics and have lifetime equivalent to the scope in which they are declared as most variable types. Class type variables, on the other hand, are handled by reference and can either be manually instantiated and deleted or have an automatic memory management strategy which handles a class instance allocation and deletion. Moreover, as they use reference semantics, it is possible for several class variables to reference a single class instance.

There are several other relevant smaller features available in Chapel. Some, like error handling and iterators, are common in more recent programming languages but absent or burdensome to use in traditional HPC languages. Others are not that common but particularly convenient for scientific computing such as native support for composite variable types i.e. complex numbers, tuples and range types.

4. NUMERICAL EXPERIMENTS

4.1 Ringleb Flow

The Ringleb flow simulation consists of an isentropic flow with an analytic solution for the Euler equations which can be derived from the hodograph transformation (Shapiro, 1953). Since this case has an analytic solution, the numerical solution error can be easily calculated, making this an interesting problem for accuracy assessment and computational cost evaluation. The parameters for the case considered here are based on the test case from the International Workshop on High-Order Methods in CFD (Wang *et al.*, 2013), such that the results can be compared to each other. The flow depends on the inverse of the stream function, k , and the velocity magnitude, v_t . In the present simulations, these parameters are chosen as $k = 0.7$ and $k = 1.5$, in order to define the bounding streamlines, and $v_t = 0.5$ to define the inlet and outlet boundaries. An interesting property of the Ringleb flow test case is that it presents a shockless transition from supersonic to subsonic flow regime.

The convergence to the analytic solution depends heavily on the accuracy of the representation of the limiting streamlines. The boundary condition of a completely tangent flow is only an accurate model of the problem if the geometric boundary sits exactly on top of an analytic streamline. The boundary condition enforcement approach adopted here is to replace the tangency condition with locally computed analytic solutions, for the conserved properties, at the limiting streamlines. Therefore, even if the boundary is not in the correct location or at the exact angle, the enforced conditions correctly represent the problem.

Five, progressively refined, meshes are used in this analysis, all provided by the International Workshop on High-Order Methods in CFD (Wang *et al.*, 2013). The mesh is composed of quadrilateral cells disposed in a structured way, despite

the unstructured mesh treatment by the solver. All of these meshes are composed of quartic (4th-degree) cells which can provide an accurate geometric representation with a reduced number of elements. The high-order mesh can only be visualized when the SPs and FPs are added because the visualization tools typically used in CFD are still developing high-order visualization capabilities. Figure 1 presents some of the various meshes used to solve the Ringleb flow problem. Depending on the degree of the FR scheme, each mesh cell and edge in the original mesh is subdivided by the solution and flux points within each element in order to properly visualize the mesh and solution.

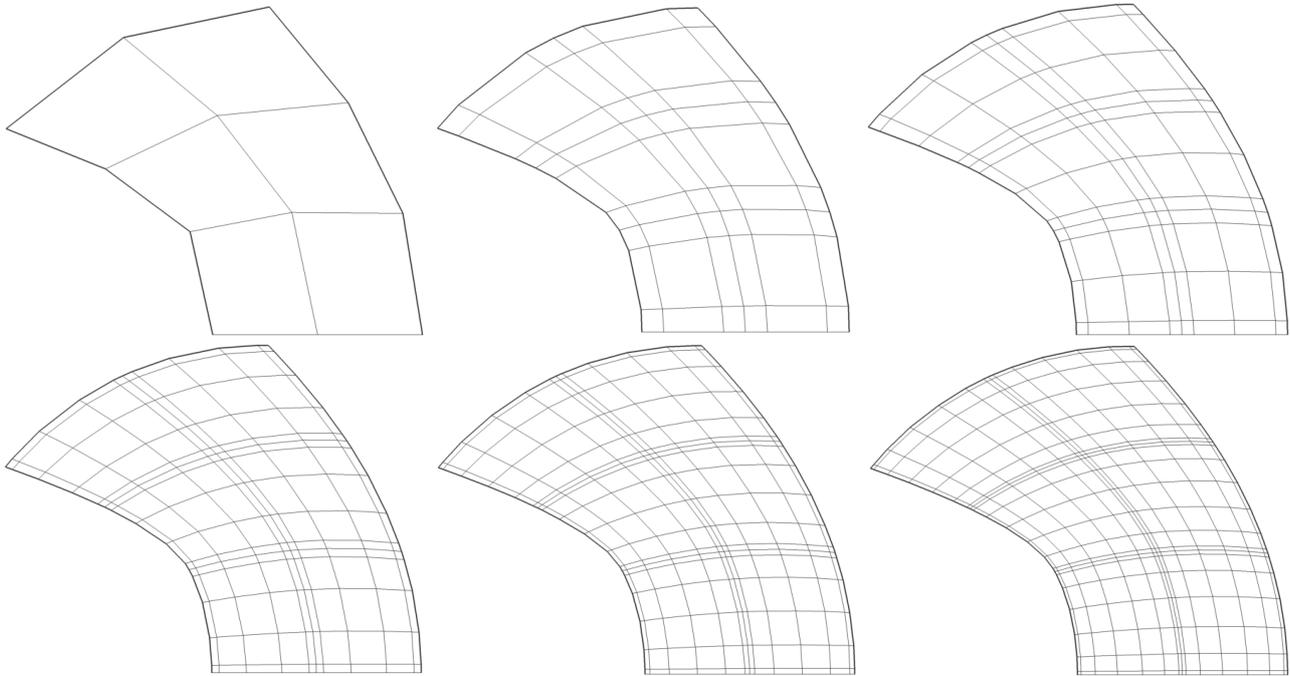


Figure 1. Top half of the coarsest original mesh and approximate representations of the effective meshes with cells subdivided by SPs and FPs from FR schemes with solution reconstructions from first to fifth degree.

This problem was simulated with 1st to 5th degree (P1 to P5) reconstruction FR schemes and the explicit 2nd-order, 3-stage Runge-Kutta time marching scheme described previously in subsection 2.2. The results are used to verify whether the code accurately solves the 2D Euler equations and achieves the expected order of accuracy for each scheme. Figure 2 presents Mach number contours for the solutions obtained with schemes using 1st, 2nd, 3rd and 4th degree polynomial reconstructions on the same meshes presented before. It should be noticed that this is an extremely coarse grid with only 12 cells and 48, 108, 192 and 300 degrees of freedom in each of the solutions shown. The 1st degree reconstruction solution has severe disturbances and asymmetry in the Mach number contours, while the second and third degree reconstruction schemes show significant improvement. The solution for the scheme with fourth order property reconstruction is visually indistinguishable from the analytic solution and, therefore, no higher degree reconstruction solution is shown here.

To calculate the solution error, the L_2 norm of the entropy error in each SP is used to numerically integrate the square of the entropy error in each cell. These are then added and finally divided by the area of the domain to obtain an average L_2 error as defined in the guidelines of the International Workshop on High-Order Methods in CFD (Wang *et al.*, 2013). The results for all schemes and meshes tested can be seen in Tab. 1 along with a "point to point" order of accuracy estimation calculated comparing the results on one mesh with a one level coarser mesh. In Fig. 3, the errors are plotted along with power function regressions using all 5 solutions for a given scheme that estimates more reliably the order of accuracy of each scheme.

The minimum expected order of accuracy for a p -th degree polynomial reconstruction scheme would be $p + 1$, which all schemes tested achieve, showing that the schemes are implemented correctly and able to solve the proposed model equations. Nonetheless it might seem surprising that the schemes surpass this expected order of accuracy by a significant margin. This can be explained by a known superconvergence characteristic of the DG, and therefore FR-DG scheme, on regular and specially on Cartesian grids (Cockburn *et al.*, 2002).

4.2 Subsonic Flow Over NACA0012 Airfoil

Another relatively simple but more physically relevant test case from the International Workshop on High-Order Methods in CFD (Wang *et al.*, 2013) is a subsonic inviscid flow over a NACA0012 profile. This test stresses the high-

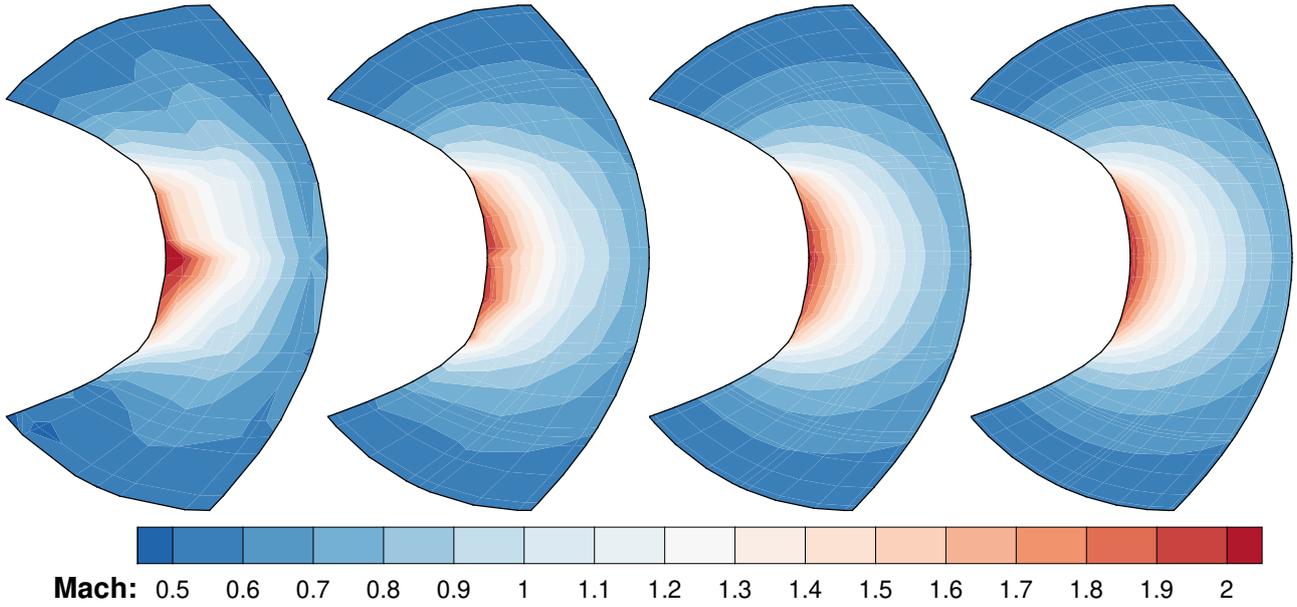


Figure 2. Mach number contours on the coarsest mesh using first to fourth degree reconstruction FR schemes.

Table 1. Ringleb flow problem errors for the FR schemes with different degrees of polynomial reconstruction for the conserved properties.

FR Scheme Solution Polynomial Degree	Cells	#DoF	$H=1/\sqrt{\#DoF}$	L_2 - Error	Order of Accuracy of the Scheme
1st	12	48	1.44×10^{-1}	3.05×10^{-4}	-
	48	192	7.22×10^{-2}	4.35×10^{-5}	2.81
	192	768	3.61×10^{-2}	5.29×10^{-6}	3.04
	768	3072	1.80×10^{-2}	6.30×10^{-7}	3.07
	3072	12288	9.02×10^{-3}	7.56×10^{-8}	3.06
2nd	12	108	9.62×10^{-2}	2.86×10^{-5}	-
	48	432	4.81×10^{-2}	1.16×10^{-6}	4.62
	192	1728	2.41×10^{-2}	4.61×10^{-8}	4.65
	768	6912	1.20×10^{-2}	2.15×10^{-9}	4.42
	3072	27648	6.01×10^{-3}	1.06×10^{-10}	4.34
3rd	12	192	7.22×10^{-2}	2.09×10^{-6}	-
	48	768	3.61×10^{-2}	2.20×10^{-8}	6.33
	192	3072	1.80×10^{-2}	6.23×10^{-10}	5.38
	768	12228	9.02×10^{-3}	8.19×10^{-12}	6.25
	3072	49152	4.51×10^{-3}	1.15×10^{-13}	6.17
4th	12	300	5.77×10^{-2}	1.84×10^{-7}	-
	48	1200	2.89×10^{-2}	7.11×10^{-9}	4.69
	192	4800	1.44×10^{-2}	3.28×10^{-11}	7.76
	768	19200	7.22×10^{-3}	1.10×10^{-13}	8.21
	3072	76800	3.61×10^{-3}	5.25×10^{-16}	7.72
5th	12	432	4.81×10^{-2}	6.01×10^{-8}	-
	48	1728	2.41×10^{-2}	2.17×10^{-10}	8.11
	192	6912	1.20×10^{-2}	1.74×10^{-12}	6.96
	768	27648	6.01×10^{-3}	9.37×10^{-16}	10.86
	3072	110592	3.01×10^{-3}	1.18×10^{-18}	9.63

order mesh representation of the airfoil surface, and has larger meshes such that performance evaluations can be made more easily. The simulations are performed with the airfoil at a 2 deg angle of attack and $M_\infty = 0.5$. The profile is defined by a slightly modified equation so that the trailing edge collapses smoothly, instead of the original definition which has a thick trailing edge. In the modified equation, the coefficient of the x^4 term is 0.1036 instead of 0.1015.

The coarsest mesh provided for the workshop is composed of 140 quadrilateral cells with the airfoil being defined by

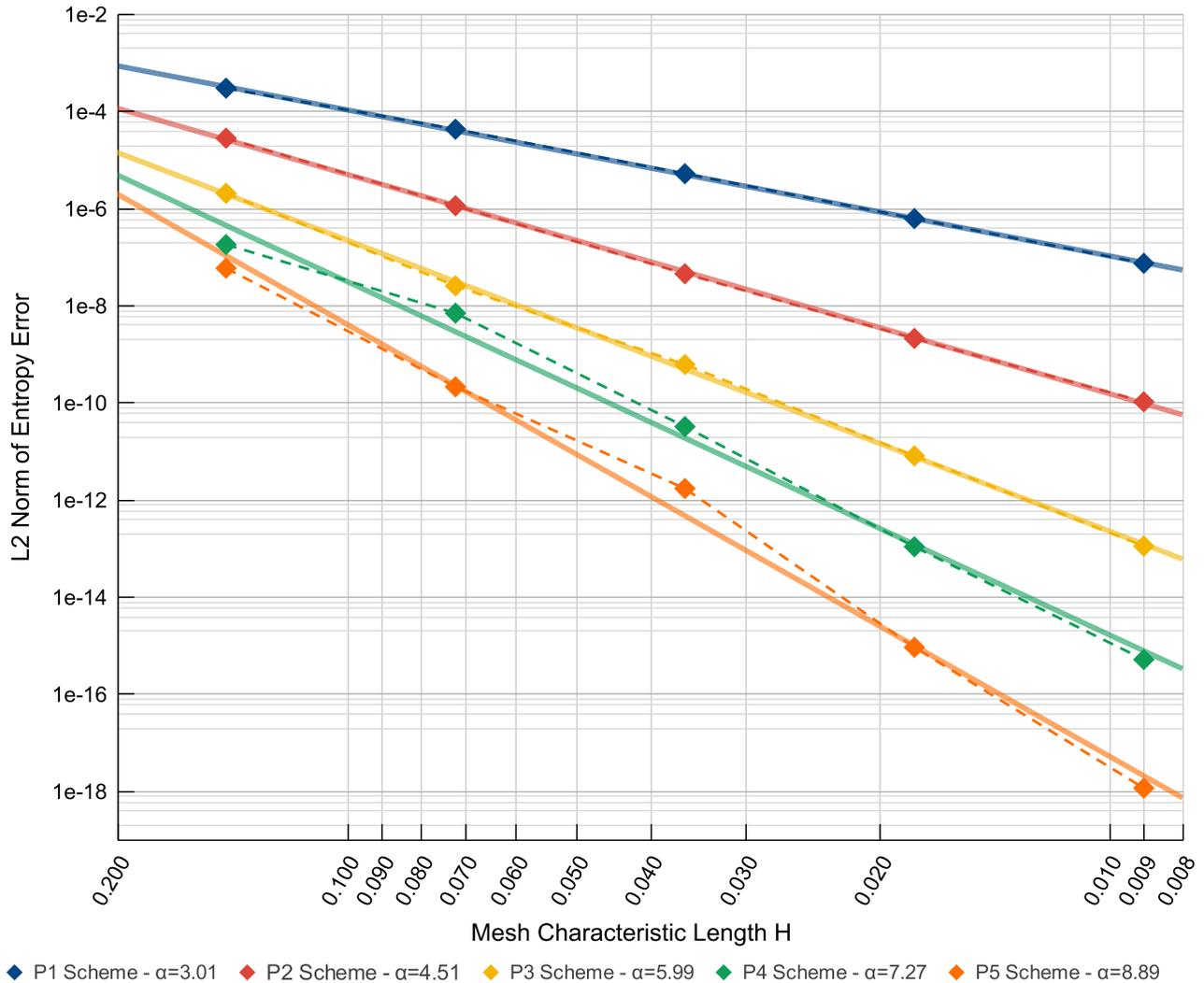


Figure 3. Plot of entropy error by mesh refinement level and regression fitted power function for FR schemes using 1st to 5th degree polynomial solution reconstructions.

10 edges. The farfield boundary is located at a distance of at least 1000 chords from the airfoil at the closest point. Five additional mesh refinement levels are provided, each with cells from the coarser mesh being subdivided into 4 new cells, generating meshes of 560, 2,240, 8,960, 35,840 and 143,360 cells, with 20, 40, 80, 160 and 320 edges defining the airfoil, respectively.

Figures 4 and 5 show a close up view of the Mach number and pressure contours obtained on the coarsest mesh, with a 4th degree reconstruction scheme. Due to the extreme coarseness of the mesh, limitations on the solution quality are clearly noticeable as a few sharp corners of the contour lines. Nevertheless the code manages to accurately capture the major features of the flow in a manner a lower order spatial discretization would not be capable of. The high-order curved mesh manages to accurately define the airfoil with only five 4th degree edges on each side so that a smooth pressure distribution is captured. While the solution quality can be improved with even higher order schemes or finer meshes, those would need to be analyzed through precise metrics, such as the entropy error, to accurately assess the improvements.

The simulations presented in this work were run on an 12 core workstation where a strong scaling test with an intermediary mesh was also performed. The 2240 cell NACA0012 mesh was iterated through for 100 time-steps with schemes using 1st to 5th degree (P1 to P5) reconstruction. The results of this test are presented in Fig. 6 along with the linear regression curves and their slope for each scheme. For most cases, the speedup measured was slightly higher than the ideal, which would be equal to the number of cores used, likely due to more efficient cache utilization. The super-linear effect seems to fall off on the tests using higher core counts which could be explained either by diminishing returns of any cache effect or possibly by memory bandwidth limitations.

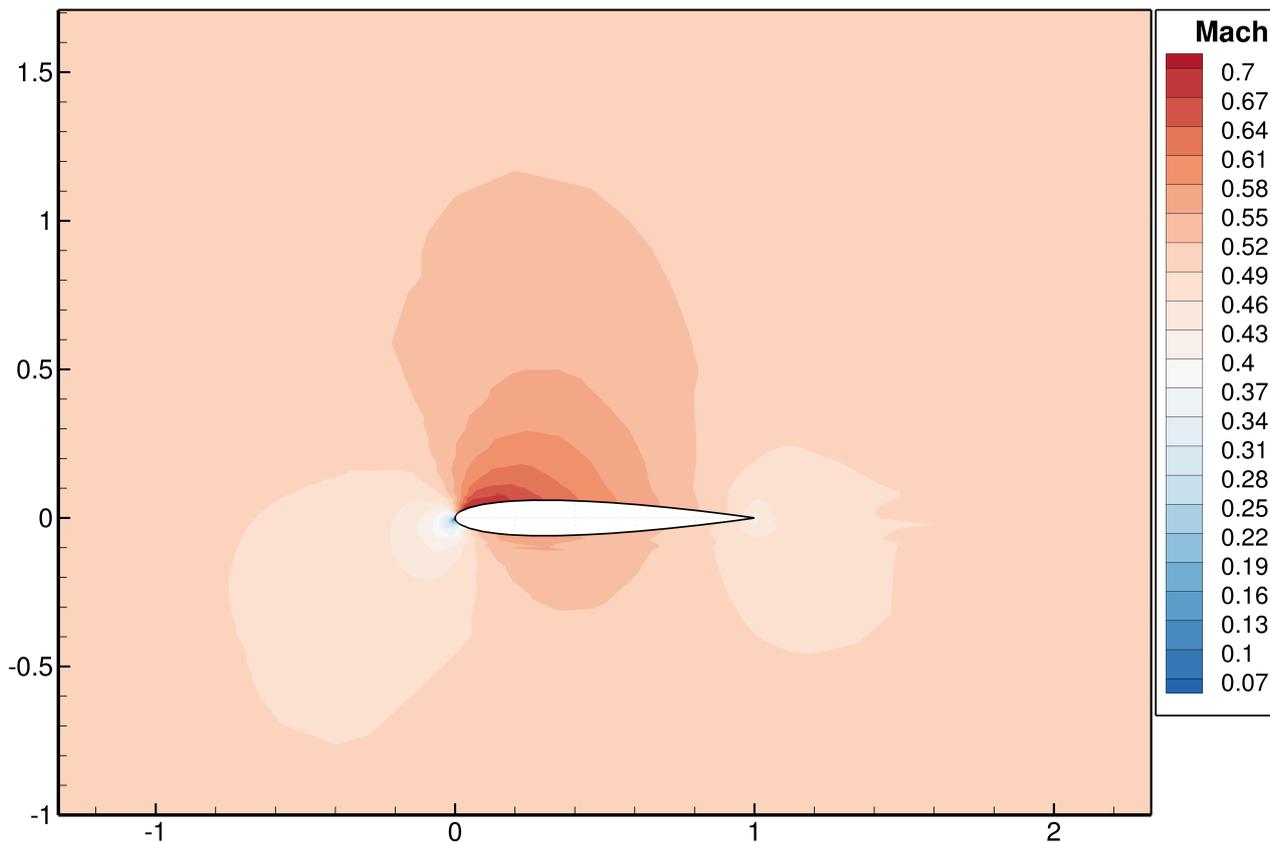


Figure 4. Mach number contours obtained on the coarsest mesh with a fourth degree reconstruction scheme.

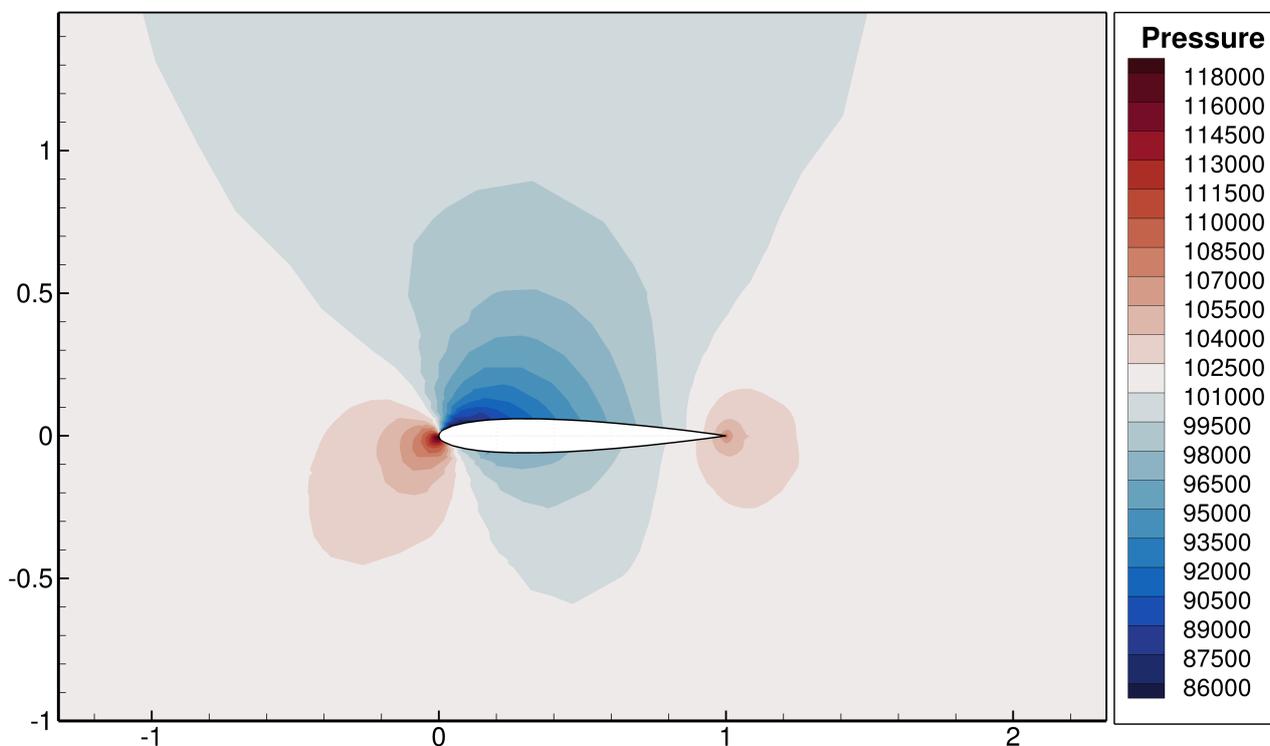


Figure 5. Pressure contours obtained on the coarsest mesh with a fourth degree reconstruction scheme.

5. CONCLUDING REMARKS

This paper presents the initial results of the development of a new CFD solver using the Flux Reconstruction (FR) method for high-order spatial discretization and the Chapel programming language, mainly for its parallel computing

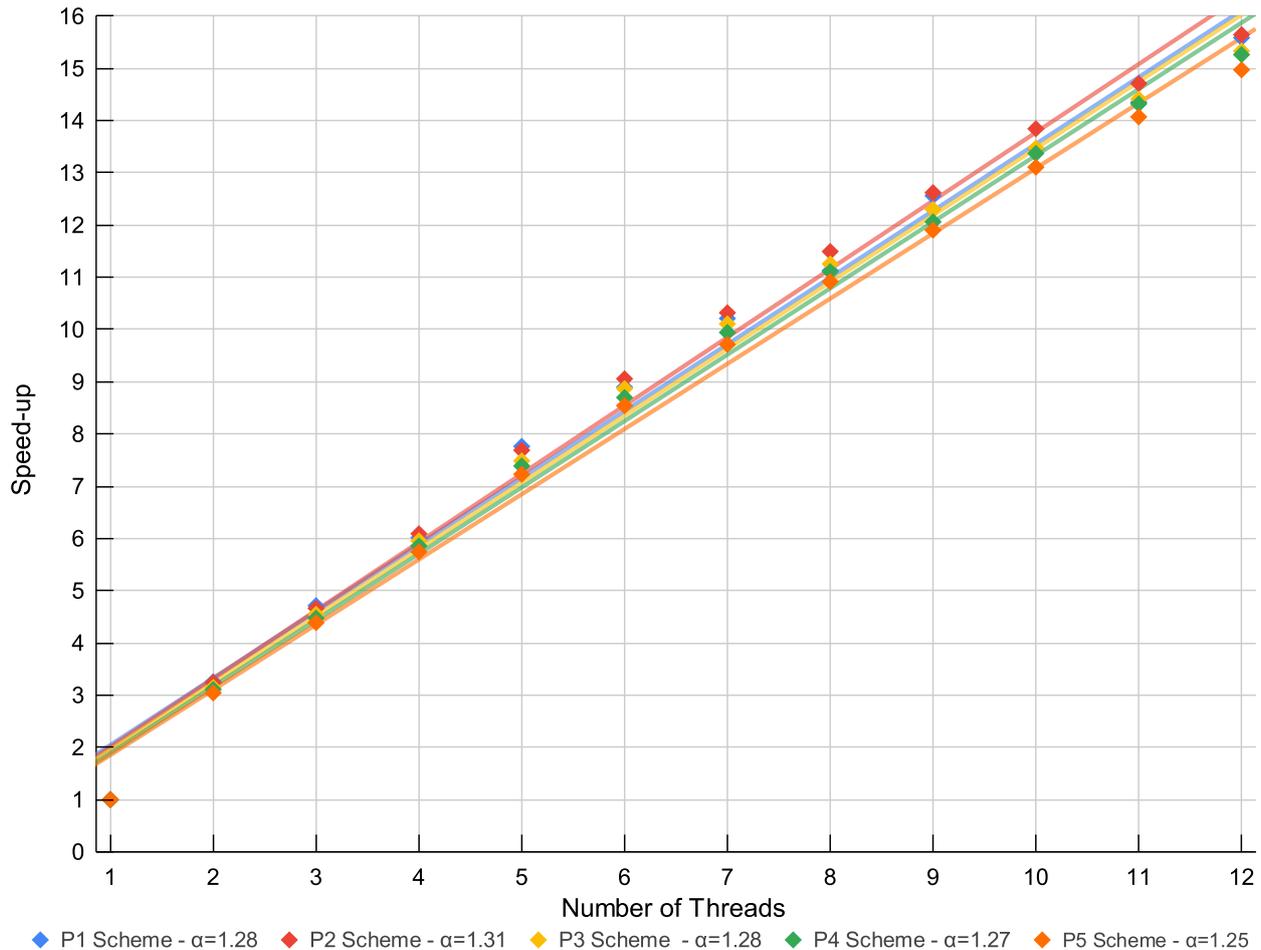


Figure 6. Strong scaling test performed on the 3rd mesh, containing 2240 cells, with 1st to 5th degree property reconstruction schemes.

features. An evaluation performed using the Ringleb flow problem shows the numerical implementation of the FR method to be correct, sometimes even exceeding the order of accuracy expected for a given scheme. A simulation of the flow over a NACA0012 airfoil on a very coarse mesh validated the high-order mesh capabilities of the code. The same test case, with a finer mesh, was also used for a strong scaling test in which the code scaled ideally from 1 to 12 cores on a single processor workstation. Subjectively, the development experience with Chapel met the authors' expectations. It enabled increased expressiveness and productivity by providing several resources through standard modules. It also required little effort to parallelize the main parts of the code. Considering these encouraging results, work to extend the code to 3D flows and distributed memory computing systems is already underway.

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support for this research provided by Fundação de Amparo à Pesquisa do Estado de São Paulo, FAPESP, under the Research Grant No. 2013/07375-0. The present work has also received support from Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, under the Research Grant No. 309985/2013-7.

7. REFERENCES

- Castonguay, P., 2012. *High-Order Energy Stable Flux Reconstruction Schemes for Fluid Flow Simulations on Unstructured Grids*. Ph.D. Thesis, Stanford University.
- Cockburn, B., Kanschat, G., Perugia, I. and Schötzau, D., 2002. "Superconvergence of the local discontinuous Galerkin method for elliptic problems on Cartesian grids". *SIAM Journal on Numerical Analysis*, Vol. 39, No. 1, pp. 264–285.
- Harten, A., 1983. "High resolution schemes for hyperbolic conservation laws". *Journal of Computational Physics*, Vol. 49, pp. 357–393.
- Huynh, H.T., 2007. "A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods". In *18th AIAA Computational Fluid Dynamics Conference*. AIAA Paper No. 2007-4079, Miami, FL.

- Huynh, H.T., 2009. “A reconstruction approach to high-order schemes including discontinuous Galerkin for diffusion”. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*. AIAA Paper No. 2009-403, Orlando, FL.
- Huynh, H.T., 2011. “High-order methods including discontinuous Galerkin by reconstruction on triangular meshes”. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. AIAA Paper No. 2011-44, Orlando, FL.
- Roe, P.L., 1981. “Approximate Riemann solvers, parameter vectors, and difference schemes”. *Journal of Computational Physics*, Vol. 43, No. 2, pp. 357–372.
- Shapiro, A.H., 1953. *The Dynamics and Thermodynamics of Compressible Fluid Flow*. Wiley, New York.
- Spiteri, R.J. and Ruuth, S.J., 2003. “A new class of optimal high-order strong-stability-preserving time discretization methods”. *SIAM Journal on Numerical Analysis*, Vol. 40, No. 2, pp. 469–491.
- Wang, Z.J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H.T., Kroll, N., May, G., Persson, P., van Leer, B. and Visbal, M., 2013. “High-order CFD methods: Current status and perspective”. *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, pp. 811–845.
- Wang, Z.J. and Gao, H., 2009. “A unifying lifting collocation penalty formulation including the discontinuous Galerkin, spectral volume/difference methods for conservation laws on mixed grids”. *Journal of Computational Physics*, Vol. 228, pp. 8161–8186.

8. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.