Verification and Application of the Unified Coordinate System in Preliminary Design

C. Nathan Woods[*] and Ryan P. Starkey[†]

University of Colorado, Boulder, Colorado, 80309

Although computational fluid dynamics has revolutionized aerodynamic analysis, it has remained less useful for preliminary design because of the difficulties associated with generating a computational mesh. Hui's unified coordinate system has the potential to alleviate this concern by virtue of its automatically generated mesh. In this work, the grid convergence of the unified coordinate system is presented. The method is found to converge, albeit at a lower rate than fixed-mesh methods. Shock-induced instabilities are found, and their effects on convergence are documented. Additionally, it is found that strong expansion waves can lead to regions of decreased grid resolution downstream of the expansion. Despite these challenges, the unified coordinate system is found to produce results that are more accurate than those from traditional methods for some problems, allowing the use of coarse meshes where high resolution would otherwise be required. Combined with the ability to automatically generate a computational mesh, the unified coordinate system is a very attractive choice for certain problems, particularly design problems where the benefits of computational fluid dynamics are outweighed by the costs of traditional grid generation methods.

Nomenclature

$\mathbf{E}$	vector of conserved variables
$\mathbf{F}$	conservative ξ flux vector
$\mathbf{G}$	conservative η flux vector
A	ξ component of x $(\frac{\partial x}{\partial \xi})$
B	ξ component of y $(\frac{\partial y}{\partial \xi})$
L	η components of x $(\frac{\partial x}{\partial \eta})$
M	η component of y $(\frac{\partial y}{\partial \eta})$
I	velocity component in the ξ direction
J	velocity component in the η direction
T	intermediate variable
S	intermediate variable
e	specific energy
g	grid point velocity function
h	grid/flow velocity ratio
p	pressure
q	magnitude of flow velocity
t	cartesian temporal coordinate
u	velocity component in the x direction
v	velocity component in the y direction
x	physical spatial coordinate
y	physical spatial coordinate
Δ	Jacobian of coordinate transformation

[*]Ph.D. Student, Aerospace Engineering Sciences, Student Member AIAA.
[†]Assistant Professor, Aerospace Engineering Sciences, Associate Fellow AIAA.

λ	computational temporal coordinate
ξ	computational spatial coordinate
η	computational spatial coordinate
θ	flow angle
ρ	mass density

I. Introduction

Computational fluid dynamics, or CFD, dramatically altered the field of aerodynamics during the twentieth century, and continues to be an essential part of aerodynamic analysis around the world, both as a supplement to wind-tunnel testing and as a replacement when circumstances require. Unfortunately, CFD is not without its costs. Computing a fluid-dynamical solution for a complicated flow problem can be slow, the equations solved are only approximations to the actual physical system, and every simulation must start with a computational mesh, the form of which can drastically affect overall solution accuracy and computational efficiency. Because of these limitations, CFD remains largely the province of highly trained professionals and academics, whose education and experience are integral in obtaining accurate results.

The dependence of CFD simulations on the computational mesh used is particularly troubling, as a mesh generated by an unskilled user can produce highly erroneous results. This problem has spawned decades of study, and many good methods for generating these meshes have been found, but most of these methods require active participation and understanding from the user. For a complicated flow that must be resolved with a high degree of certainty, generation of the mesh can take weeks, even months. As a result, accurate CFD has been limited to applications where the up-front costs of mesh generation by competent specialists are justified.

Unfortunately, those applications do not include early-stage projects and preliminary design work. When simulation boundaries regularly change, as is common during the early stages of vehicle design, each change necessitates the generation of a new mesh, the costs of which rapidly outweigh the benefits CFD simulations provide. As a result, critical design decisions are made without the highly relevant aerodynamic information that CFD can provide, and bad decisions made early on can be very difficult to undo later.

The goal of this current work is not to improve computational speed, but rather to enable the generation of reasonably accurate solutions to complex flow problems at low user-involvement and low computational cost. The Unified Coordinate System (UCS) defined by W. H. Hui[1] shows promise in this area, but its characteristics have never been fully explored and documented. In this paper, we present the results of a formal grid-convergence study on the original implementation of the UCS method.[2] We further characterize its behavior, which will provide direction to those looking to use UCS in their projects. Several interesting test cases are also presented to further highlight the utility of the method.

II. Background on the Unified Coordinates

The unified coordinate system arose from a series of studies that used Lagrangian coordinates to clearly resolve slip lines in supersonic flows, and it is best understood in that context. In the current implementation, fluid nodes move with the fluid, as in the Lagrangian case, but the speed of the fluid nodes is merely proportional to the speed of the fluid, varying by a constant factor h that can vary from node to node. The additional freedom introduced by this scalar function in space allows the simulation to capture many useful properties of the Lagrangian coordinate system, while also maintaining a regular (orthogonal, if the flow permits) grid and without any changes in connectivity of nodes. In particular, the grid follows fluid streamlines for steady flows, automatically conforms to solid boundaries, and resolves slip lines nearly perfectly. It does this at the cost of requiring the dynamic creation and destruction of nodes at in- and out-flow simulation boundaries, and the computational costs associated with solving a linear, first order, partial differential equation in space at each time step.

II.A. The UCS transformation

A detailed description of the mathematics involved is given by Hui,[2] and is summarized as follows.

A generic, time-dependent, coordinate transformation in three dimensions can be written

$$\begin{pmatrix} dt \\ dx \\ dy \\ dz \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ U & A & L & P \\ V & B & M & Q \\ W & C & N & R \end{pmatrix} \begin{pmatrix} d\lambda \\ d\xi \\ d\eta \\ d\zeta \end{pmatrix}. \tag{1}$$

If we set U, V, W to hu, hv, hw, where u, v, and w are the Cartesian components of the fluid velocity, and h is an arbitrary scalar function whose values lie between 0 and 1, then the points of our transformation move in the same direction as fluid particles, at a lower speed. h can be chosen as a constant value, and $h = 0$ and $h = 1$ correspond to the traditional Eulerian and Lagrangian coordinate systems, respectively. Alternatively, h can be constrained to yield more desireable grid properties. In particular, it is possible to use h to prevent severe mesh distortion. For two-dimensional flows, the mesh can even be made orthogonal. Although methods for doing so will be discussed, for the purposes of this paper h will be taken as a constant value unless otherwise specified.

II.B. Development and application of the unified coordinates

The unified coordinates were first used to solve time-dependent flows in 1999,[2] where they were applied to the two-dimensional, inviscid, Euler equations for an ideal gas. In that paper, a Godunov scheme with a flux-limited MUSCL extension to second order was used to solve a variety of problems, including a steady Riemann problem, flow through a transonic duct, Mach reflection of a traveling shock wave, and an implosion/explosion problem. In 2001,[3] a similar scheme was applied in three dimensions to the steady Riemann problem and to supersonic flow past a corner. Two-dimensional, inviscid, external flows around both steady and oscillating airfoils were presented later.[4]

UCS has been applied to more than just the Euler equations. An application to the full Navier-Stokes equations[5] showed that UCS was capable of resolving viscous flows, but also that the system was robust enough to handle complex phenomena like shock-induced boundary-layer separation with recirculation, thus answering one of the major concerns about its Lagrangian origins. Additional models to which UCS has been applied include reactive flows,[6] multi-material flows,[7] magneto-hydrodynamics,[8] and gas-kinetic BGK simulations of a freely falling plate[9].[10]

UCS provides several advantages over traditional CFD, such as better slip line resolution and streamline-oriented grids, but our interest in the system arises primarily from its automatic generation of a computational mesh. Design optimization codes will produce many different design options (thousands), and there is a need for a low-level CFD program that is reasonably fast, reliably accurate, and requires little interaction on the part of the user to validate these designs. The automatically generated, streamline-aligned coordinate system UCS provides has the potential to satisfy these requirements, if it can be shown to work reliably and in an intuitive way for a wide range of simulations with verifiable results.

III. The Two-dimensional Euler Equations in Unified Coordinates

The two-dimensional Euler equations in unified coordinates are given by Hui,[2] and are summarized as follows:

$$\frac{\partial \mathbf{E}}{\partial \lambda} + \frac{\partial \mathbf{F}}{\partial \xi} + \frac{\partial \mathbf{G}}{\partial \eta} = 0 \tag{2}$$

where

$$\mathbf{E} = \begin{pmatrix} \rho\Delta \\ \rho\Delta u \\ \rho\Delta v \\ \rho\Delta e \\ A \\ B \\ L \\ M \end{pmatrix}, \mathbf{F} = \begin{pmatrix} \rho(1-h)I \\ \rho(1-h)Iu + pM \\ \rho(1-h)Iv - pL \\ \rho(1-h)Ie + pI \\ -hu \\ -hv \\ 0 \\ 0 \end{pmatrix}, \mathbf{G} = \begin{pmatrix} \rho(1-h)J \\ \rho(1-h)Ju - pB \\ \rho(1-h)Jv + pA \\ \rho(1-h)Je + pJ \\ 0 \\ 0 \\ -hu \\ -hv \end{pmatrix} \tag{3}$$

and

$$\Delta = AM - BL,\ I = uM - vL,\ J = Av - Bu \tag{4}$$

where Δ represents the Jacobian of the transformation, and is equivalent to the physical volume of each computational grid cell. I and J are the $\xi-$ and $\eta-$ components of the fluid velocity, respectively. The parameter h can be chosen to preserve grid angles, thus ensuring that an initially orthogonal mesh remains orthogonal for all time. Since the presence of stagnation points can potentially lead to singularities in h, it is better to solve for $g = \ln\left(h\sqrt{u^2+v^2}\right)$, which is the natural log of the actual grid speed required in order to preserve orthogonality, and remains bounded, even when h is singular. The resulting equation for g is given by

$$\alpha(\xi,\eta)\frac{\partial g}{\partial \xi} + \beta(\xi,\eta)\frac{\partial g}{\partial \eta} + \gamma(\xi,\eta) = 0 \tag{5}$$

where

$$\alpha = \left(L^2+M^2\right)^2 (A\cos\theta - B\sin\theta) \tag{6}$$

$$\beta = \left(A^2+B^2\right)^2 (M\cos\theta - L\sin\theta) \tag{7}$$

$$\gamma = -\left(L^2+M^2\right)^2 (A\cos\theta + B\sin\theta)\frac{\partial\theta}{\partial\xi} + \left(A^2+B^2\right)^2 (L\cos\theta + M\sin\theta)\frac{\partial\theta}{\partial\eta} \tag{8}$$

and θ is the flow angle given by $u = q\cos\theta$ and $v = q\sin\theta$. If this equation is solved independently of the fluid flow at each time step, then it is a first-order, linear partial differential equation, and can be solved in a variety of ways. In particular, if the coefficients α, β, and γ do not change sign, then the equation can be solved by the method of characteristics, as well as more advanced techniques such as adomian decomposition and variational iteration. Iterative relaxation schemes are also effective, even if the coefficients do change sign.

As Eq. 2 is in strong conservation form, it can be solved by established shock-capturing methods. We shall use the Godunov scheme with MUSCL refinement given by Hui.[2] It should be noted that the unified coordinate system requires a system of eight equations, rather than the four in Eulerian coordinates, however the additional equations are very simple, and do not significantly increase computational cost of the simulation. The solution of the equation for g is more complicated, and Hui reports that overall execution time in unified coordinates is typically increased by about 10% over Eulerian methods.[1]

IV. Simulation Details

IV.A. Spatial discretization

The spatial discretization scheme used here is more reminiscent of the smoke and schlieren imaging techniques of experimental fluid dynamics than of a traditional body-fitted mesh. A column of evenly spaced nodes is created at upstream simulation boundaries, and these nodes flow through the simulation region much as particles would, with nodes entering and leaving the simulation region as needed. The motion of these nodes is governed by Eq. 1. These nodes contain all of the information necessary for the simulation: the flow variables p, ρ, u, and v; the metric components of the transformation; the grid motion parameter h (and equivalently, g); the physical location of the node x and y; and finally knowledge of the connectivity with neighboring nodes or boundaries. The use of Eq. 5 can ensure that neighboring nodes remain neighbors throughout the simulation, thus obviating the need for connectivity changes.

IV.B. Flow solver and time discretization

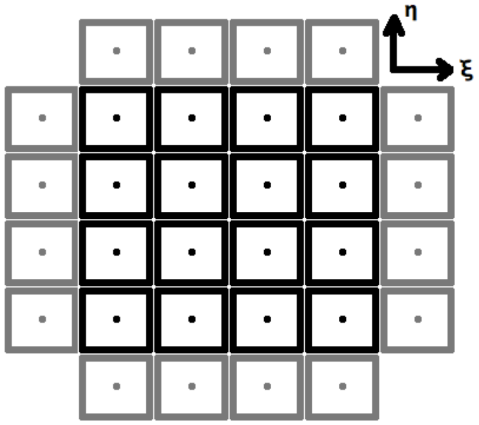


Figure 1. Diagram of computational ξ-η space. Ghost cells are shown in gray.

As mentioned above, Eq. 2 can be solved by any shock-capturing scheme, and we follow Hui in choosing the Godunov scheme with a MUSCL update to second-order in space. The Riemann problem is solved as in Hui[2] to calculate the flow variables at cell interfaces for each cell. As there is no Riemann solution to the full two-dimensional problem, we must make two additional approximations. First, the equation set is artificially decoupled into parts: the conservation equations for p, ρ, u and v form the first group; the geometric conservation equations for A, B, L, and M form the second group; and the equation for h forms the third. These equation groups are then solved in turn to advance the solution a full time step. For the second approximation, a first-order Strang dimensional splitting scheme is applied in order to reduce the Riemann problem for each cell to a collection of one-dimensional problems. That is, the differential operator $\mathcal{L}_{\xi\,\eta}^{\Delta\lambda}$ is approximated as $\mathcal{L}_{\xi}^{\Delta\lambda/2}\mathcal{L}_{\eta}^{\Delta\lambda}\mathcal{L}_{\xi}^{\Delta\lambda/2}$.

The algorithm is given as follows. For each time step:

1. Compute optimal time step.[5] Predict the maximum coordinate advancement of the first column of nodes. If necessary, limit the time step such that the first column ends the time step no more than $\Delta\xi$ from the upstream boundary, to prevent nonuniformity in the node spacing.

2. Apply Strang splitting, as above:

   (a) Identify the appropriate time-advancement and the active interfaces for the given Strang step. Active interfaces are left/right for the ξ steps of Strang splitting, and top/bottom for the η step.

   (b) Step through nodes. For each node $n_{i,j}$:

      i. Compute values of flow variables at both active cell interfaces:

         A. Identify adjoining states using neighboring nodes, boundary conditions, and MUSCL interpolation, as appropriate.

         B. Solve the local, one-dimensional Riemann problem to obtain the values of flow variables at the interface between adjoining states.

      ii. Use interfacial flow values and the value of h that corresponds to the cell being updated to compute updated cell metric coefficients A, B, L, and M. Store all updated node values separately until after all nodes have been computed.

      iii. Compute physical flux into the cell using the values of flow variables at the interfaces and the updated geometric variables corresponding to the cell being updated.

      iv. Use flux to compute updated flow values.

   (c) Update all nodes.

3. Solve Eq. 5 for h and update coordinate positions using trapezoidal integration.

4. Add/remove columns of nodes as needed.

By artificially decoupling the equations that govern the evolution of the flow from those that govern the metric, the conservation equations (Eq. 2) reduce to the Euler equations in arbitrary curvilinear coordinates, together with separate, very simple evolution equations for the metric coefficients, and an equation for computing h.

It is also possible to solve the riemann problem exactly for the one-dimensional UCS equations by decoupling only the equation for h and making additional approximations,[3] but the end resulting algorithm is identical in either case.

IV.C. Application of boundary conditions

Boundary conditions are implemented through the use of ghost cells, which are then solved within the Godunov framework just as any other cell. The flow state within the ghost cells is chosen according to Table 1.

Table 1. Application of boundary conditions

Boundary Type	Specified Quantity(s)	How Enforced
Supersonic Inflow	Pressure, Density, Velocity, Metric, h/g	Entire flow state is directly specified.
Supersonic Outflow	(none)	Flow state is copied from simulation interior
Solid Walls	Flow Angle	Flow state is copied from simulation interior. Velocity and metric are reflected across wall angle.
Constant Pressure	Pressure	Flow state is copied from simulation interior. Pressure is specified.

The first-order Godunov method requires only the knowledge of directly adjacent cells. For the application of the second-order MUSCL update however, it is necessary to have knowledge of flow states two cells away in each direction. For cell interfaces that correspond to simulation boundaries, the MUSCL update is neglected. This has no effect on the application of boundary conditions, which are satisfied at the interface regardless. For interior cells, the boundary condition is applied normally whenever the MUSCL update reaches beyond the boundary.

IV.D. MUSCL refinement

As the Godunov scheme is only first-order accurate in time, a flux-limiting, linear, MUSCL interpolation is used to improve the accuracy to second-order.[11] The specific implementation used here is given by Hui and reproduced below.[2]

$$\begin{gathered} f_r = f_{i+1,j} - 0.5\left(f_{i+2,j} - f_{i+1,j}\right)\phi\left(r^+\right) \\ r^+ = \frac{\left(f_{i+1,j} - f_{i,j}\right)}{\left(f_{i+2,j} - f_{i+1,j}\right)} \\ f_l = f_{i,j} + 0.5\left(f_{i,j} - f_{i-1,j}\right)\phi\left(r^-\right) \\ r^- = \frac{\left(f_{i+1,j} - f_{i,j}\right)}{\left(f_{i,j} - f_{i-1,j}\right)} \\ \phi\left(r\right) = \max\left(0, \min\left(1, r\right)\right) \end{gathered} \tag{9}$$

IV.E. Controlling node motion

Hui recommends that, for flows in which the coefficients α and β change sign, the solution of Eq. 5 be solved by iteration. This has been implemented in the simplest way possible in this work, using a first-order successive over-relaxation scheme. If the coefficients do not change sign, it can also be done using the method of characteristics or variational iteration.

IV.F. Coordinate advancement

At the end of each time step, the x and y locations of each node must be updated. Trapezoidal integration is both simple and effective, maintaining second-order accuracy at low computational cost.

$$x_{i,j}^{n+1} = x_{i,j}^{n} + \frac{1}{2}\left(h_{i,j}^{n}u_{i,j}^{n} + h_{i,j}^{n+1}u_{i,j}^{n+1}\right)\Delta\lambda \tag{10}$$

$$y_{i,j}^{n+1} = y_{i,j}^{n} + \frac{1}{2}\left(h_{i,j}^{n}v_{i,j}^{n} + h_{i,j}^{n+1}v_{i,j}^{n+1}\right)\Delta\lambda \tag{11}$$

V. Code Verification

In order to verify the UCS method, a systematic, grid-convergence study has been performed for problems meant to fully exercise the method and for which exact solutions are known. First, a steady, supersonic Riemann problem is solved, the solution of which consists of an oblique shock, an expansion fan, and a slip line. Next, a wall-induced obique shock wave and a Prandtl-Meyer expansion are also tested.

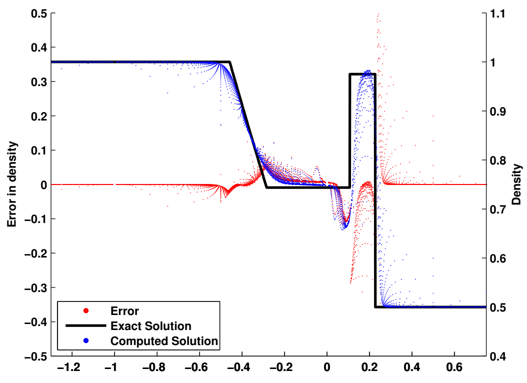
V.A. The Riemann problem and order of accuracy

In order to perform a reliable verification, it is important to stress the solver. The Riemann problem given by Hui,[2] with its three distinct nonlinear waves, provides such a stress. It is given by the upstream boundary condition:

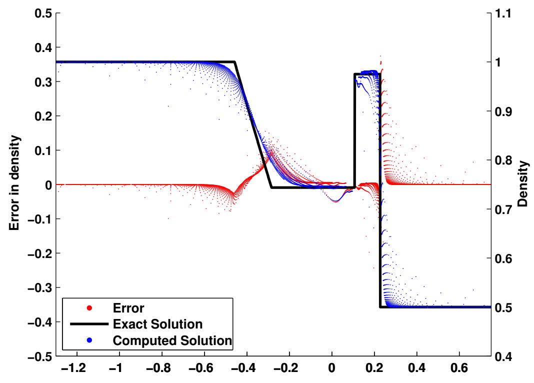
$$(p, \rho, M, \theta) = \begin{cases} (0.25, 0.5, 7, 0) & : y > 0 \\ (1, 1, 2.4, 0) & : y < 0 \end{cases} \tag{12}$$

where M is the Mach number and θ is the flow angle.

The solution to this problem contains a shock, an expansion wave, and a slip line, which come together at a singularity at the origin. Figs. 2(a) and 2(b) demonstrate the greatly improved resolution of the slip line when using unified coordinates as compared to the traditional Eulerian coordinate system, and reveal the sacrifice of accuracy near the expansion fan.



(a) $h = 0$



(b) $h = 0.999$

Figure 2. The similarity solution of the Riemann problem and the corresponding error in the numerical solution, computed throughout the simulation region.

Fig. 3 quantifies these findings. The solution is shown to converge in three different test configurations, albeit with lower convergence rates than were expected. It is known that monotone finite difference methods such as the plain Godunov method, are 1/2-order accurate overall.[12] The UCS scheme is formally second-order based on the minmod limiter, and therefore should be at least 5/8-order accurate.[13] In the results shown here, the observed convergence rates are markedly lower. There are several possible explanations for this behavior:

- When the computational mesh is moving, the singularity at the upstream boundary is imperfectly resolved, and its apparent location may vary as cells move by.
- Boundary conditions in general are applied using only first-order accurate methods.
- The error of the time-step-eulerian approximation is unknown.
- The Strang dimensional splitting approximation used here is only first-order.

V.B. The oblique shock and induced instabilities

In order to further verify the method, we run a test problem for an oblique shock wave. The upstream condition is given by

$$(p, \rho, M, \theta) = (1, 1, 1.8, 0) \tag{13}$$

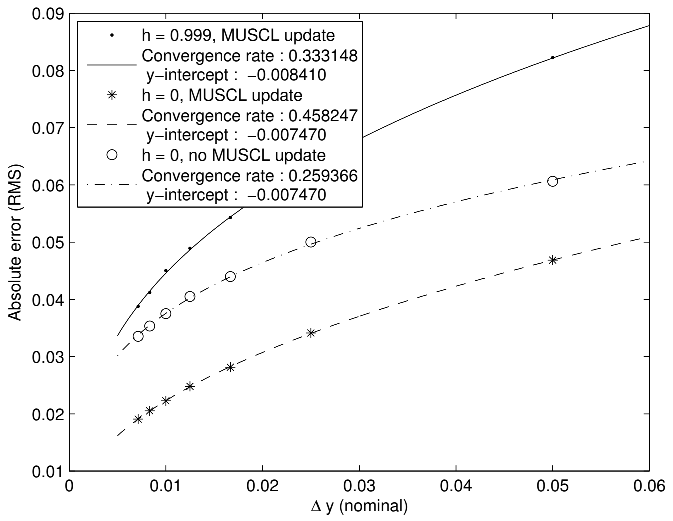
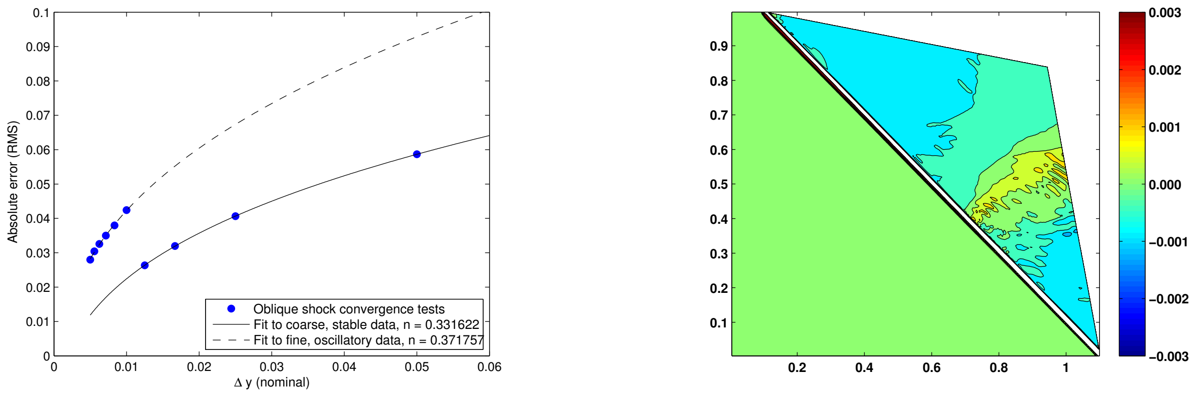


Figure 3. Root-mean-squared error for the riemann problem, with order of convergence n.

and the wall angle that induces the shock is chosen to give a downstream flow angle of 45^o. The grid was generated automatically, with a value of $h = 0.999$.



(a) Root-mean-squared error for the oblique shock problem. The appearance of grid instabilities leads to two distinct error curves with different rates of convergence n.

(b) A plot of normalized error in pressure, highlighting the oscillations which propogate downstream from the oblique shock.

Figure 4. The oblique shock wave

The root-mean-squared error of the computed solution for the oblique shock closely follows the behavior exhibited by that of the Riemann problem, with one major caveat. As the grid is refined, there is a critical refinement level beyond which oscillations appear in the solution, as seen in Fig. 4(b). These oscillations increase the error, though the solution continues to converge.

V.C. The expansion fan and “grid separation”

A final verification test is run for a Prandtl-Meyer expansion corner. Convergence is again demonstrated (see Fig. 5), but the expansion corner reveals an important limitation of the UCS method. In the presence of a strong expansion wave, the decreased grid resolution can cause the computational nodes to pull away from the wall, as can be seen in Fig. 6. Thus, although the streamlines eventually do align with the wall as expected, the dramatic loss of resolution near the wall can seriously affect any further downstream results.

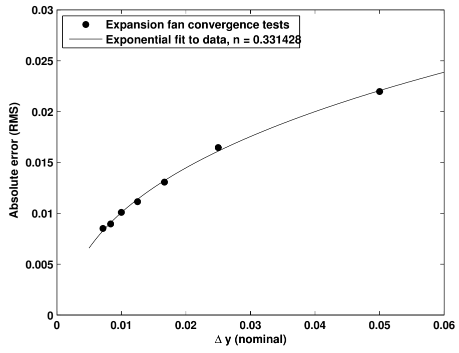


Figure 5. Root-mean-squared error for the Prandtl-Meyer expansion, with order of convergence n.

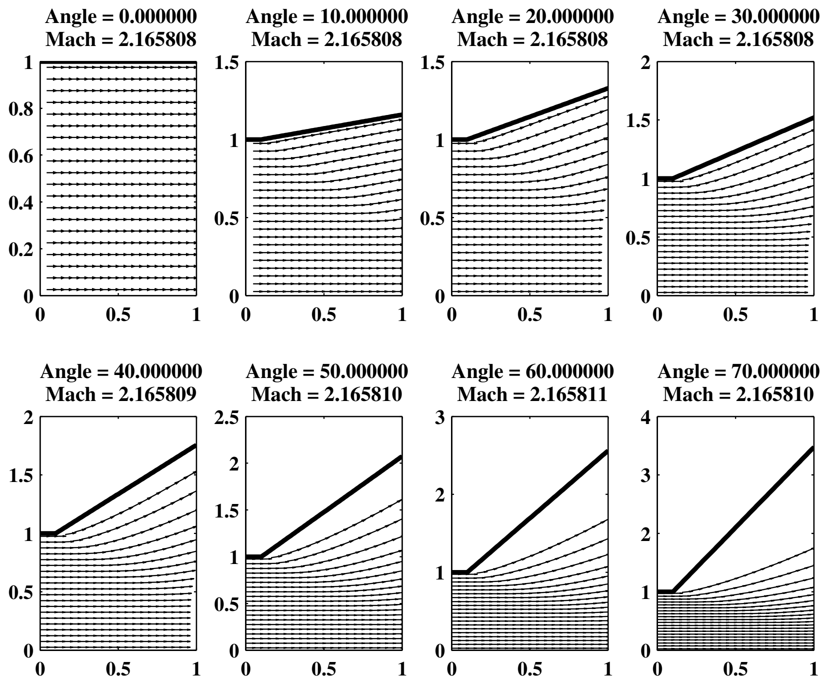


Figure 6. Computed streamlines for Prandtl-Meyer expansion at increasing expansion angles. Angles are given in degrees.

VI. The Unified Coordinates in Preliminary Design

The preceeding verification has detailed instabilities and concerns associated with the unified coordinates, but UCS has many significant advantages over traditional CFD which make it very attractive for certain applications.

The very first order of project design is typically based on an algebraic description of the system being modeled. This description is, by default, crude and approximate, but it can be solved extremely quickly, making it possible to solve it thousands of time while iterating through different design possibilities. The designer is then left with a dozen or so options that satisfy the algebraic description of the system to be designed. Going beyond that level of fidelity in fluid dynamics requires a CFD simulation. The automatically generated, flow-aligned mesh UCS provides simplifies this initial application of CFD, allowing designers to further limit the field to only a few designs for which the added effort of generating a high-quality mesh for truly accurate CFD is warranted.

VI.A. Pressure boundary conditions and nozzle plume flows

The unified coordinate system is especially useful for problems where the physical boundaries themselves are unknown, such as pressure boundary conditions. Consider the nozzle plume flows in Fig. 7, which were generated with constant pressure boundary conditions with no more difficulty than a parallel-wall channel. The mesh itself simply flows to fill the streamtube defined by the nozzle, freeing the user from defining exactly where boundaries lie. This is in contrast to traditional methods, where an estimate would have to be made of the maximum diameter of the nozzle streamtube, and the simulation sized to fit around that maximum diameter.

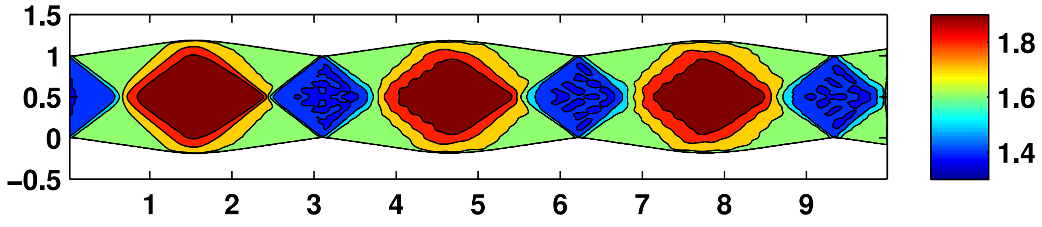


Figure 7. Computed Mach number for an under-expanded nozzle flow, showing the diamond-shock train. Note the presence of oscillations which grow as the flow progresses downstream.

VI.B. Isentropic nozzle flow

Flow around boundaries of arbitrary shape and curvature are also easy to solve. Flow through a parabolic, convergent nozzle was modeled, again using only the automatically generated mesh (Fig. 8). The upstream flow conditions are given by:

$$(p, \rho, M, \theta) = (0.1081, 0.2596, 2.3, 0.0) \tag{14}$$

The profile of the nozzle is given as a piecewise function of derivatives:

$$f'(x) = \begin{cases} 0 & : 0 < x < 1 \\ -0.015625x + 0.03125 & : 1 < x < 3 \\ 0.015625x - -0.15625 & : 3 < x < 7 \\ -0.015625x + 0.28125 & : 7 < x < 9 \\ 0 & : 9 < x < 10 \end{cases}$$

For boundaries of this type, care must be taken. As in the case of the expansion corner in Fig. 6, the computational mesh can pull away from the physical boundary if the curved expansion is too strong.

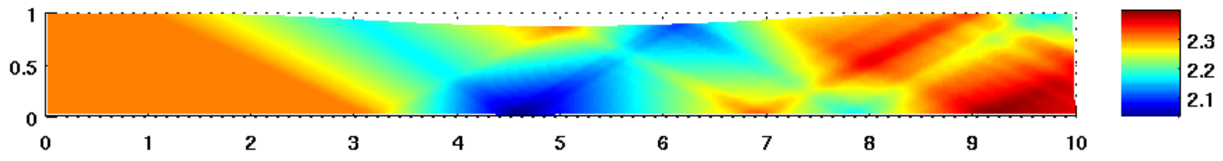


Figure 8. Computed Mach number for isentropic flow through a constricting, parabolic, supersonic channel

VI.C. A transonic duct flow

Finally, we use the UCS method to model flow through a convergent duct. This problem is identical to the one given by Hui,[2] and is characterized by the formation of a mach stem and the resulting transonic flow (Fig. 9). The mesh again flows through the duct, conforming to solid walls without need for user specification of the mesh.

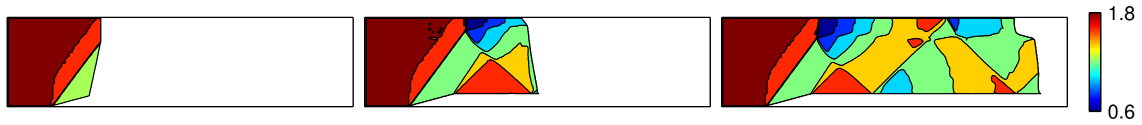


Figure 9. Computed Mach number for transonic duct flow at various times, showing the manner in which nodes flow to fill boundaries.

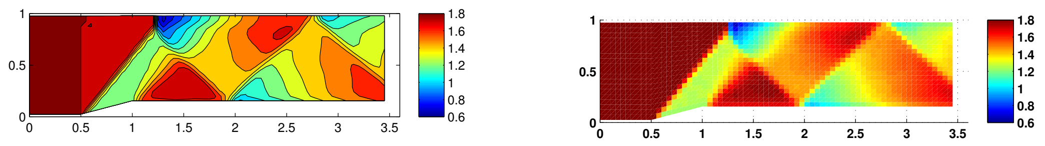
It can be seen in Fig. 10 that the Eulerian scheme ($h = 0$) fails to capture the formation of the slip line at low grid resolutions, and also shows slightly less accurate prediction of shock locations when compared with a much higher resolution solution. The UCS method, on the other hand, does resolve the slip line, and makes slightly more accurate predictions of shock locations, at the cost of a less-well-resolved expansion corner.

VII. Conclusion

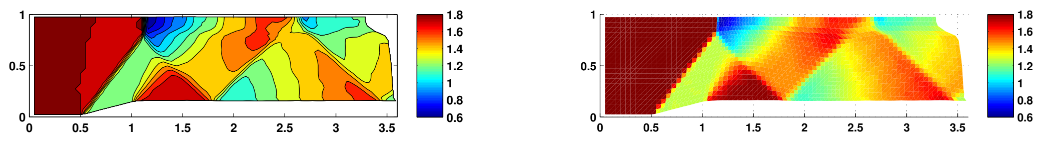
The unified coordinate system continues to show promise as a tool for preliminary designers to validate low-order designs. We have used a formal grid-convergence study to verify the UCS methodology for the two-dimensional Euler equations, a necessary step in the maturation of the scheme. UCS is found to converge to exact solutions less quickly than a similar, shock-capturing, fixed-mesh method. Additionally, numerical instabilities were uncovered in the presence of shocks, and the scheme behaves unexpectedly near strong expansion corners. UCS does show better accuracy at comparable grid resolution for some complex problems. The ease of use associated with UCS, particularly its automatically generated mesh, makes it well suited for validating preliminary design work, especially when problems are small and user time is comparatively more valuable than computational time.

VIII. Future Work

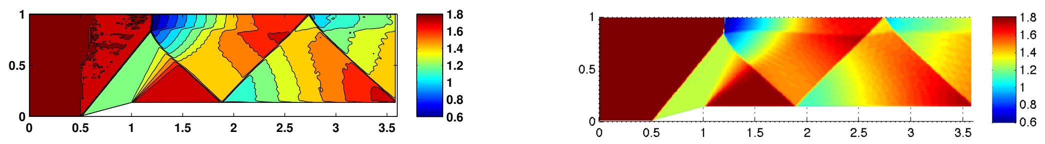
The current implementation of UCS must be modified in order to improve numerical stability and to enforce conservation. There also exists the possibility of extending UCS in order to further improve its usefullness for preliminary design. Since the scheme already resolves slip lines very well through its flow-oriented mesh, it is natural to investigate whether it is possible to simultaneously resolve shocks the same way, thus greatly improving local solution accuracy in regions with the highest potential stresses. Additionally, the ease with which UCS adapts to curved boundary conditions suggests the incorporation of a boundary-layer solver to account for viscous effects, and work on such a solver is already underway.



(a) $h = 0$, nominal grid size: 72×20



(b) $h = 0.25$, nominal grid size: 72×20



(c) $h = 0$, nominal grid size: 360×100

Figure 10. Qualitative accuracy comparison between UCS (b) and Eulerian (a,c) simulations for a transonic duct flow. Notice the improved resolution of the slip line and the walls for the UCS solution.

References

[1]Hui, W. H., "The Unified Coordinate System in Computational Fluid Dynamics," *Communications in Computational Physics*, Vol. 2, No. 4, August 2007, pp. 577–610.

[2]Hui, W. H., Li, P. Y., and Li, Z. W., "A Unified Coordinate System for Solving the Two-Dimensional Euler Equations," *Journal of Computational Physics*, Vol. 153, 1999, pp. 596–673.

[3]Hui, W. H. and Kudriakov, S., "A Unified Coordinate System for Solving the Three-Dimensional Euler Equations," *Journal of Computational Physics*, Vol. 172, January 2001, pp. 235–260.

[4]Hui, W., Hu, J., and Zhao, G., "Gridless Computation Using the Unified Coordinates," *Computational Fluid Dynamics 2004*, edited by C. Groth and D. W. Zingg, Springer Berlin Heidelberg, 2006, pp. 503–508.

[5]Hui, W., Wu, Z. N., and Gao, B., "Preliminary Extension of the Unified Coordinate System Approach to Computation of Viscous Flows," *Journal of Scientific Computing*, Vol. 30, No. 2, February 2007, pp. 301–344.

[6]Azarenok, B. N. and Tang, T., "Second-order Godunov-type scheme for reactive flow calculations on moving meshes," *Journal of Computational Physics*, Vol. 206, No. 1, 2005, pp. 48 – 80.

[7]Jia, P., Jiang, S., and Zhao, G., "Two-dimensional compressible multimaterial flow calculations in a unified coordinate system," *Computers and Fluids*, Vol. 35, 2006, pp. 168–188.

[8]Zhilkin, A., "A Dynamic Mesh Adaptation Method for Magnetohydrodynamics Problems," *Computational Mathematics and Mathematical Physics*, Vol. 47, No. 11, 2007, pp. 1819–1832.

[9]Jin, C. and Xu, K., "A unified moving grid gas-kinetic method in Eulerian space for viscous flow computation," *Journal of Computational Physics*, Vol. 2007, 2007, pp. 155–175.

[10]Jin, C. and Xu, K., "Numerical Study of the Unsteady Aerodynamics of Freely Falling Plates," *Communications in Computational Physics*, Vol. 3, No. 4, April 2008, pp. 834–851.

[11]Toro, E. F., *Riemann Solvers and Numerical Methods for Fluid Dynamics, A Practical Introduction*, Springer-Verlag, 2nd ed., 1999.

[12]Sabac, F., "The Optimal Convergence Rate of Monotone Finite Difference Methods for Hyperbolic Conservation Laws," *SIAM Journal on Numerical Analysis*, Vol. 34, No. 6, 1997, pp. pp. 2306–2318.

[13]Popov, B. and Trifonov, O., "Order of Convergence of Second Order Schemes Based on the Minmod Limiter," *Mathematics of Computation*, Vol. 75, No. 256, 2006, pp. 1735 – 1753.