A Unified CFD Approach to High-Speed Component Design

C. Nathan Woods* and Ryan P. Starkey†

University of Colorado, Boulder, Colorado, 80309

Computational fluid dynamics has traditionally been difficult to apply to early-stage projects and design work, because it has been a complex field requiring a high degree of competence from those who work in it, and small, new projects typically have not had dedicated fluid dynamicists available. The unified coordinate system offers an accurate, mesh-generation-free, intuitive method for solving fluid mechanical problems quickly, without much involvement on the part of the users. The basics of one unified coordinate scheme are presented, and examples taken from various aspects of propulsion design are used to discuss these advantages in detail.

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

*Ph.D. Student, Aerospace Engineering Sciences, Student Member AIAA.

†Assistant Professor, Aerospace Engineering Sciences, Associate Fellow AIAA.

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 accuracy, 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.

In order to reduce the level of training required for CFD, we are exploring the possibilities offered by Hui's unified coordinate system.[1] As a result of its roots in Lagrangian fluid dynamics, UCS offers several key advantages that could allow for accurate CFD solutions, with no computational mesh generation, in a way that is both simple to use and intuitive to understand and analyze. The current work uses preliminary examples to show how UCS can accomplish all of these goals in a way that will not require undue sacrifices of computational speed and scalability.

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 an arbitrary scalar function h. By its very nature, this Lagrangian-esque coordinate system may provide all of the features identified previously as requirements for design-oriented CFD, at least for steady flows. The mesh flows into the computational region, and the body-fitted mesh is generated all on its own. The mesh so generated naturally provides good resolution of both shocks and especially slip lines. Finally, the results of computations on the flowing mesh are intuitively sensible, even to those of limited knowledge and experience in fluid mechanics. This will be shown through examples hereafter, following an introduction to the unified coordinates.

II.A. The UCS transformation

A detailed description of the unified coordinate system 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, such as preventing 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 flow of a two-dimensional, inviscid, 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]

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

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

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

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

where

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

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

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

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

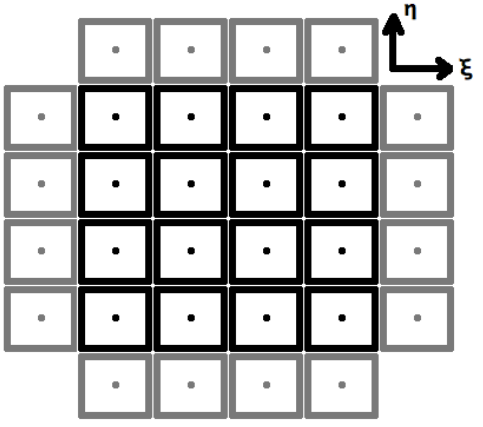


Figure 1. Diagram of computational ξ-η space. Ghost cells are shown in gray.

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 $\mathcal{L}^{\Delta\lambda}_{\xi\,\eta}$ is approximated as $\mathcal{L}^{\Delta\lambda/2}_{\xi}\mathcal{L}^{\Delta\lambda}_{\eta}\mathcal{L}^{\Delta\lambda/2}_{\xi}$.

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]

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

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. The Unified Coordinates in Design

As was previously mentioned, the Lagrangian nature of the unified coordinate system allows for accurate CFD solutions, automatic mesh generation, and a scheme that is simple, straightforward, and intuitive in its behavior. We present as examples a two-dimensional, steady Riemann problem; a transonic channel flow; a diamond shock train; flow into a variable supersonic inlet; and viscous flow through a constant-area channel.

V.A. Resolution and formal convergence

The steady Riemann problem is chosen because it has an exact, discontinuous solution from which formal convergence can be calculated. The initial condition is given by:

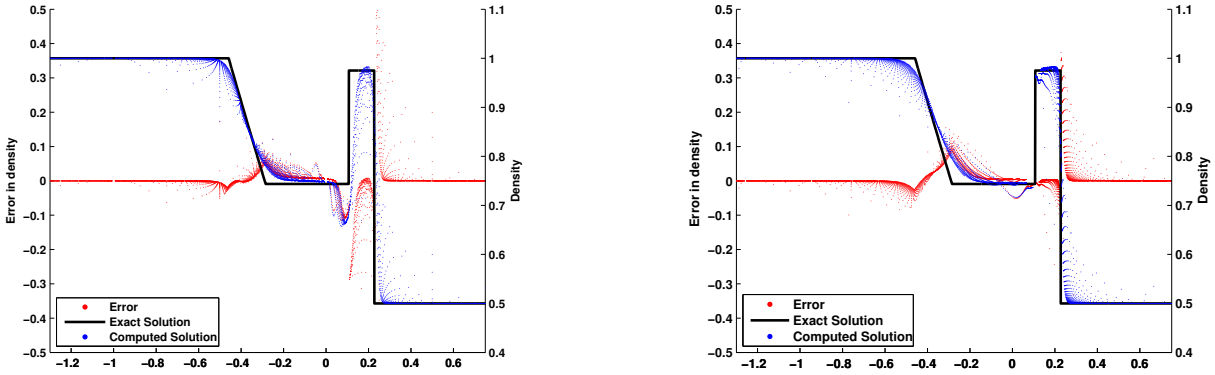
$$(p, \rho, M, \theta) = \begin{cases} (0.25, 0.5, 7, 0) & : \frac{y}{x} > 0 \\ (1, 1, 2.4, 0) & : \frac{y}{x} < 0 \end{cases} \tag{12}$$

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

The solution is:

$$(p, \rho, M, \theta) = \begin{cases} (0.25, 0.5, 7, 0) & : \frac{y}{x} > 0.22732 \\ (0.660558, 0.975061, 5.90382, 0.107478) & : \frac{y}{x} > 0.107894 \\ (0.660558, 0.743644, 2.6671, 0.107478) & : \frac{y}{x} > -0.284151 \\ (1, 1, 2.4, 0) & : \frac{y}{x} < -0.458349 \end{cases} \tag{13}$$

Figs. 2(a) and 2(b) show both the exact and computed solutions as well as the difference between them for both a stationary mesh and a moving one. The improvement in slip line resolution is clearly apparent, and there is some improvement along the shock, as well, due to the tendency of computational nodes to cluster together in regions of compression.



(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.

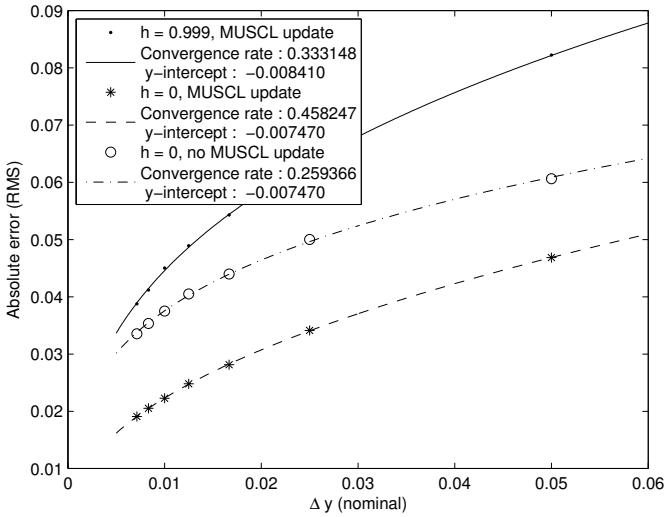


Figure 3. Root-mean-squared error for the riemann problem, with order of convergence n.

Fig. 3 shows the grid convergence of the current implementation of the UCS method, together with an Eulerian comparison. 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.

Further testing and development will be required in order to better understand the convergence of this scheme.

V.B. A transonic duct flow

Following Hui,[2] we use the UCS method to model flow through a convergent duct. This problem is characterized by the formation of a mach stem and the resulting transonic flow (Fig. 4). The mesh flows through the duct, filling out the solid walls as it goes (see Fig. 4). This example is particularly noteworthy because it contains both supersonic and subsonic flows, without any special treatment.

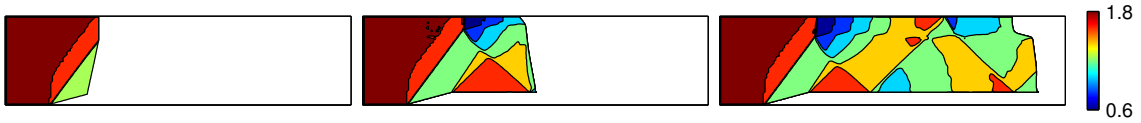
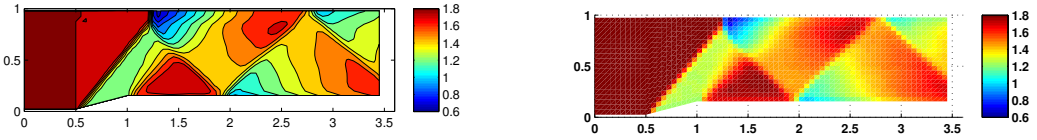
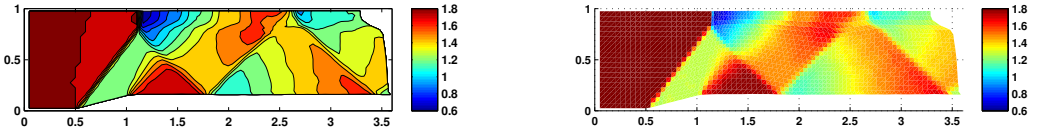


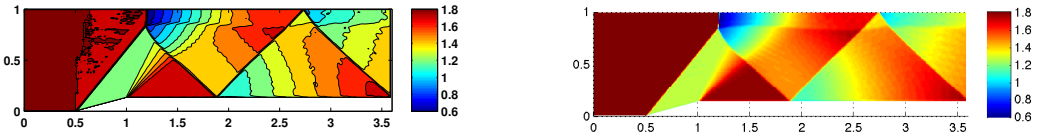
Figure 4. Computed Mach number for transonic duct flow at various times, showing the manner in which nodes flow to fill boundaries.



(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 5. 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.

This flow was also used to compare the performance of UCS against Eulerian coordinates. It can be seen in Fig. 5 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.

V.C. 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. 6, 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.

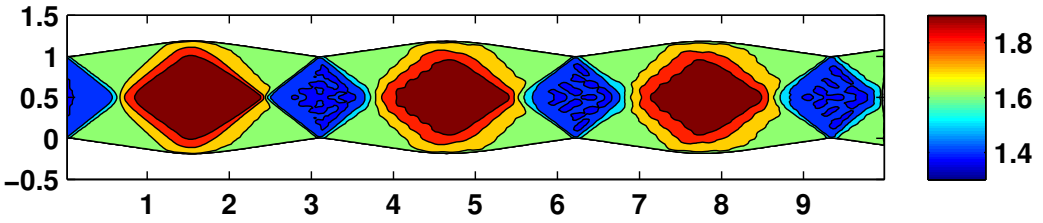


Figure 6. Computed Mach number for an under-expanded nozzle flow, showing the diamond-shock train. Note the presence of oscillations which grow as the flow progresses downstream.

V.D. Model F-14 inlet

One of the principal attractions of the UCS method is the potential to represent complex flow geometries in a simple, intuitive way and without grid generation. To better illustrate this feature, a more complex inlet model was chosen, based off of publicly available sketches of the inlet of the now retired USAF F-14. This inlet is approximately two-dimensional and is designed to provide subsonic flow to the engine at freestream Mach number in the range $0 < M < 2.3$. This is accomplished through the use of internal, variable ramps, as shown in Fig. 7.

This is exactly the kind of problem that UCS can excel at. Defining an approximate flow geometry is simple, and the automatic grid generation allows for quick solutions at different freestream conditions, and the correspondingly different inlet geometries.

Unfortunately, shock-induced instabilities have plagued this effort. It was impossible to obtain a full steady-state solution, and it was likewise impossible to resolve any subsonic or transonic flows. The full solution of this problem will therefore be contingent on a more stable solution scheme. A completely supersonic flow, having not quite come to steady-state, is shown in Fig. 8, and the growing instabilities are visible in the last frame.

V.E. Basic boundary-layer effects in a uniform channel

Many phenomena in hypersonics are a result of the interaction of the viscous boundary layer with the inviscid flow, especially in the isolator. The normal and oblique shock trains are products of the interaction between the boundary layer and the inviscid core flow in the isolator, and many other interactions besides. As a result, there is an interest in coupling this inviscid flow solver with a boundary-layer solver in order to account for viscous effects.

Although the current implementation is crude, where the boundary layer is represented using the simple turbulent, flat-plate, constant pressure formula given in Schlichting,[14] it serves as a useful proof-of-concept for the method, and will be a valuable step toward implementing an actual boundary-layer solver. In the simulation, viscous effects are included by enforcing a solid wall condition that aligns with the boundary-layer displacement thickness.

Results from one such proof-of-concept test are shown in Fig. 9. The presence of the boundary layer can be clearly seen in the curving of the inviscid flow in the otherwise uniform channel, as well as the formation of the oblique shock train.

VI. Conclusion

The present work has shown preliminary examples of two-dimensional, inviscid fluid flows that highlight the various strengths of the unified coordinate system. Accuracy and convergence were shown for the steady Riemann problem, and a two-dimensional duct flow was used to demonstrate accuracy as well. All of the problems demonstrate the automatic mesh generation characteristic of the unified coordinates, and they all display the behavior that makes the unified coordinate system both simple and intuitive for users who lack sophistication in fluid mechanics. These characteristics, together, provide an appropriate framework for CFD users who are not experts in the field. Access to this kind of simple, accurate, intuitive CFD will open

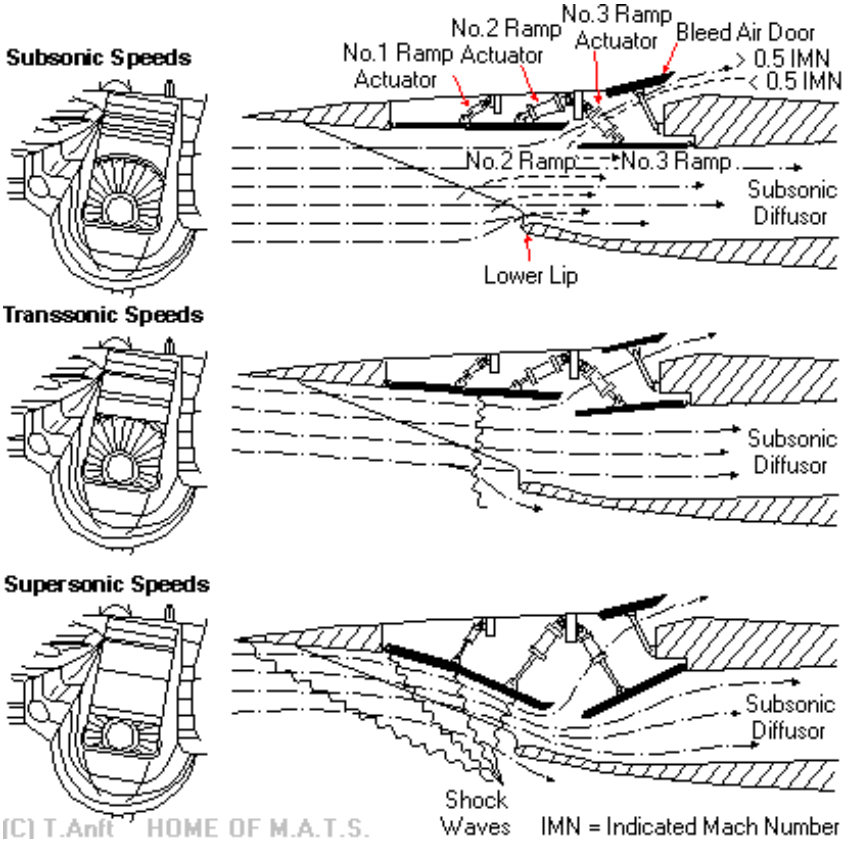


Figure 7. Diagram of the variable inlet geometry of the USAF F-14 Tomcat. Courtesy Home of M.A.T.S., http://www.anft.net/f-14/f14-detail-airintake.htm

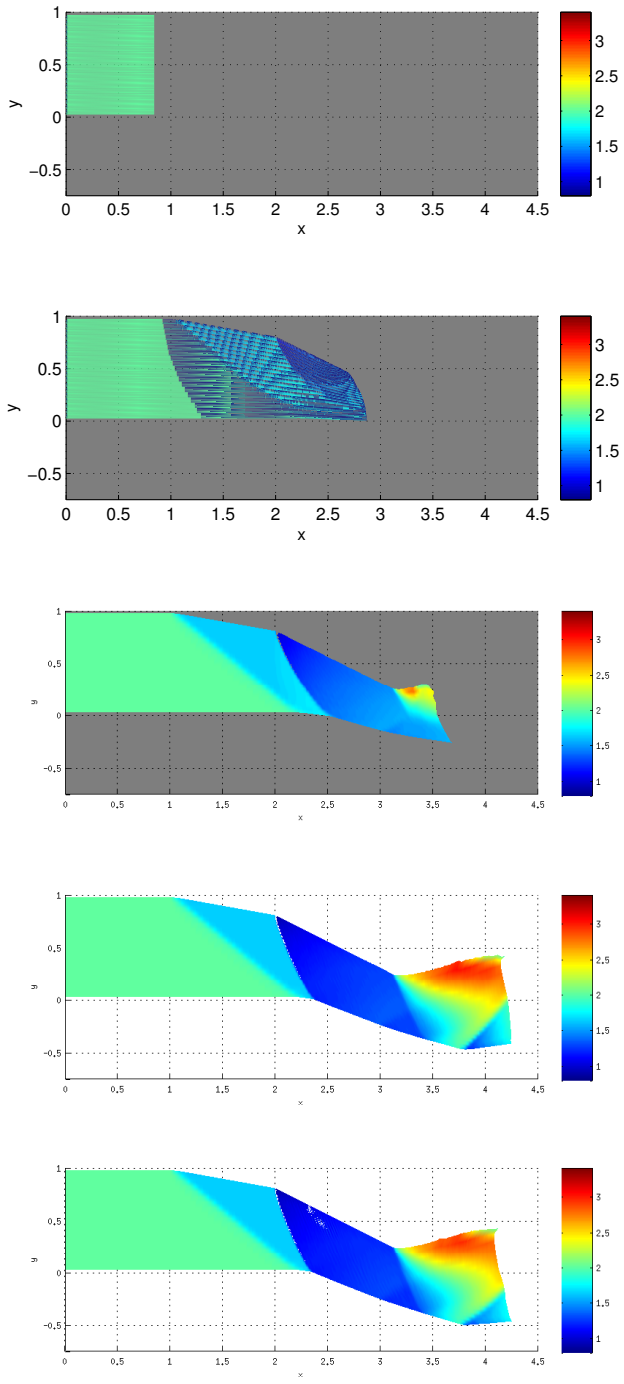


Figure 8. Time-lapse images of Mach number in a geometry modeled after Fig. 7.

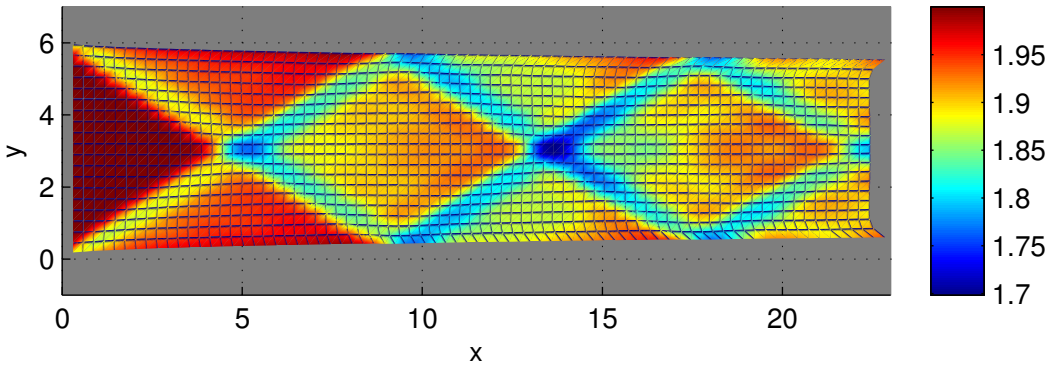


Figure 9. Oblique shock train produced by a turbulent boundary layer in an otherwise uniform channel

up the field to amateurs and designers, and may revolutionize the use of CFD in preliminary design.

VII. Future Work

The single most critical task yet to be accomplished is the stabilization of the UCS scheme. Some means must be found to eliminate shock-induced instabilities, which will open up UCS to highly compressive inlets, more general transonic flows, and long flowpaths, such as an entire hypersonic engine. Additionally, the ease with which boundary-layer effects were incorporated into the inviscid solution immediately suggests the application of a more sophisticated boundary-layer solver.

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.

[14]Schlichting, H. and Gersten, K., *Boundary Layer Theory*, Springer-Verlag, 8th ed., 2000.