Comprehensive Exam Proposal

C. Nathan Woods

November 25, 2011

Abstract

Computational fluid dynamics, or CFD, has tremendous potential for applications in high-speed-vehicle design, but this potential has been unrealized due to the high initial costs, both human and computational, of CFD simulations. The purpose of this work is to use W. H. Hui's unified coordinate system to craft an accurate, efficient, two-dimensional CFD program for compressible aerodynamics with complex geometries. This program will combine routines for inviscid flow with boundary-layer techniques and Navier-Stokes solvers to reduce computational costs using a stream-oriented, moving grid framework developed by Hui.

Contents

List of Figures

Nomenclature

A	Component of coordinate transformation, equivalent to $\frac{\partial x}{\partial \xi}$
B	Component of coordinate transformation, equivalent to $\frac{\partial y}{\partial \xi}$
C	Component of coordinate transformation, equivalent to $\frac{\partial z}{\partial \xi}$
J	Jacobian of the coordinate transformation
L	Component of coordinate transformation, equivalent to $\frac{\partial x}{\partial \eta}$
M	Component of coordinate transformation, equivalent to $\frac{\partial y}{\partial \eta}$
Ma	Mach number
N	Component of coordinate transformation, equivalent to $\frac{\partial z}{\partial \eta}$
P	Component of coordinate transformation, equivalent to $\frac{\partial x}{\partial \zeta}$
Q	Component of coordinate transformation, equivalent to $\frac{\partial y}{\partial \zeta}$
R	Component of coordinate transformation, equivalent to $\frac{\partial z}{\partial \zeta}$
U	x component of grid velocity, equivalent to $\frac{\partial x}{\partial \tau}$

U_η	η component of grid velocity, equivalent to $\frac{\partial\eta}{\partial t}$
U_ξ	ξ component of grid velocity, equivalent to $\frac{\partial\xi}{\partial t}$
U_ζ	ζ component of grid velocity, equivalent to $\frac{\partial\zeta}{\partial t}$
V	y component of grid velocity, equivalent to $\frac{\partial y}{\partial\tau}$
W	z component of grid velocity, equivalent to $\frac{\partial z}{\partial\tau}$
$\mathbf{E}, \mathbf{F}, \mathbf{G}$	Conservative variable and flux vectors
η	Component of position in computational space
$\frac{D_{\vec{u}}\eta}{Dt}$	Material derivative with respect to fluid velocity $\vec{u}$
$\frac{D_{\vec{U}}}{Dt}$	Material derivative with respect to grid velocity $\vec{U}$
κ	Grid skewness
$\mathcal{L}_{x,y}^{\Delta z}$	Differential operator that advances a solution by Δz, dependent on x and y
$\nabla_{\vec{x}}$	Gradient operator with respect to physical spatial coordinates
τ	Time in computational space
θ	Flow angle
$\vec{U}$	Grid velocity 3-vector, with Cartesian components given by U, V, W
$\vec{\xi}$	Vector of computational coordinates, including τ as the 0^{th} component
$\vec{u}$	Fluid velocity 4-vector, with 1 as the 0^{th} component
$\vec{x}$	Vector of physical, cartesian coordinates, including t as the 0^{th} component
ξ	Component of position in computational space
ζ	Component of position in computational space
$_{\alpha,\beta}$	Summation index running from 0 to 3

$_{i,j}$	Summation index running from 1 to 3
g	Modified grid motion function
h	Grid velocity proportionality function
n	Time-step index
t	Time in physical space
x	x component of position in physical space
y	y component of position in physical space
z	z component of position in physical space
δ	Bow shock standoff distance
ν	Kinematic viscosity
R_c	Radius of curvature
R_s	Bow shock radius of curvature
Re_x	Reynolds number with respect to x
u_∞	Inviscid core velocity condition for boundary-layer

1 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, especially for design work. To quote Jameson, "CFD is still not being exploited as effectively as one would like in the design process. This is partially because of the long set-up times and high costs, both human and computational, of complex flow simulations."[1]

Automatic grid generation for complex problems would greatly reduce the set-up time and human costs, and good progress has been made on this front using unstructured grids. However, structured grids, which offer significant advantages in terms of memory use, grid adaptation, and algorithm maturity (see [2][3][4] and [5]), they have so far resisted automation.

W.H Hui et al developed and demonstrated a method of automatic, structured, grid generation using a time-dependent grid whose motion was a function of the local fluid velocity. The grid is generated at the upstream boundary and allowed to flow through the simulation region until it fills the area of interest. The actual grid motion is completely specified by requiring the grid velocity to preserve some quantity of interest, such as skewness or jacobian. This eliminates the major problems of skewness and arbitrarily small jacobian[6] associated with grid-point-movement methods of solution-adaptive grid refinement, while simultaneously yielding the excellent resolution of slip lines that is characteristic of a streamline-oriented grid.

Although much has been done to develop Hui's methods in various situations, no one has ever assembled the disparate prototype solvers into a coherent whole. The goal of this project is to craft a powerful new design tool by assembling the different ideas into a complete package that will automatically and quickly compute fluid flows under arbitrary conditions.

For instance, consider the solution of flow around the Space Shuttle. It consists of a largely supersonic flow region, with a very tight and highly reactive shock and boundary layer around the leading edge of the vehicle, and a powerful wake downstream. The ideal situation would be to accept a (perhaps time-dependent) geometric definition of the Shuttle's aerodynamic shape and compute, without user intervention, a steady supersonic solution wherever possible, and subsonic, ionized, reactive, and time-dependent bubbles wherever necessary, without generating a body-fitted grid. Unfortunately, such a solver has been the target of decades of research without full success. The goal of this dissertation will be a subset of the ideal situation, combining the following solvers in an automatic way for two-dimensional flows:

- A time-marching Euler solver to handle general inviscid flows.
- A boundary-layer equation solver to introduce viscous effects for flows without separation.
- A laminar Navier-Stokes solver to handle regions of separation and recirculation.
- A steady-state, supersonic Euler solver to greatly reduce the time required for inviscid computation.

2 Background

2.1 Hui and 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, grid points move in the same direction as the fluid, but at different speeds. The additional freedom introduced by allowing the speed of the grid points to vary allows the simulation to capture many useful properties of the Lagrangian coordinate system, such as automatic grid generation and excellent slip line resolution, while also maintaining a structured, regular (orthogonal, if the flow permits) grid. In particular, the grid follows fluid pathlines, 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, ordinary differential equation in space at each time step.

2.1.1 Coordinate Transformations

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

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

Such a coordinate transformation could also be written more succinctly in index notation:

$$dx_\alpha = \frac{\partial x_\alpha}{\partial \xi^\beta} d\xi_\beta \tag{2}$$

where $\frac{\partial x_\alpha}{\partial \xi^\beta}$ are the components of the transformation matrix given in Eq. 1. The derivatives transform differently:

$$\frac{\partial}{\partial x^\alpha} = \frac{\partial \xi_\beta}{\partial x^\alpha} \frac{\partial}{\partial \xi^\beta} \tag{3}$$

That is, the derivatives transform via the transpose of the inverse of the matrix that transforms the differentials.

$$\begin{pmatrix} \frac{\partial}{\partial t} \\ \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} = \begin{pmatrix} 1 & -U_\xi & -U_\eta & -U_\zeta \\ 0 & \frac{MR-NQ}{J} & \frac{CQ-BR}{J} & \frac{BN-CM}{J} \\ 0 & \frac{NP-LR}{J} & \frac{AR-CP}{J} & \frac{CL-AN}{J} \\ 0 & \frac{LQ-MP}{J} & \frac{BP-AQ}{J} & \frac{AM-BL}{J} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial \tau} \\ \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{pmatrix}. \tag{4}$$

where we have:

$$J = \begin{vmatrix} A & L & P \\ B & M & Q \\ C & N & R \end{vmatrix} \tag{5}$$

$$\nabla_{\vec{x}}\xi \equiv \frac{MR-NQ}{J}, \frac{NP-LR}{J}, \frac{LQ-MP}{J} \tag{6}$$

$$\nabla_{\vec{x}}\eta \equiv \frac{CQ-BR}{J}, \frac{AR-CP}{J}, \frac{BP-AQ}{J} \tag{7}$$

$$\nabla_{\vec{x}}\zeta \equiv \frac{BN-CM}{J}, \frac{CL-AN}{J}, \frac{AM-BL}{J} \tag{8}$$

$$\tag{9}$$

$$U_\xi \equiv \vec{U} \cdot \nabla_{\vec{x}}\xi \tag{10}$$

$$U_\eta \equiv \vec{U} \cdot \nabla_{\vec{x}}\eta \tag{11}$$

$$U_\zeta \equiv \vec{U} \cdot \nabla_{\vec{x}}\zeta \tag{12}$$

U_ξ, U_η, and U_ζ are the components of grid velocity in the ξ, η, and ζ directions, respectively. From Eq. 1, we can also show

$$\frac{D_{\vec{U}}}{Dt} \begin{pmatrix} \xi \\ \eta \\ \zeta \end{pmatrix} = 0 \tag{13}$$

where $\frac{D_{\vec{U}}}{Dt} \equiv \frac{\partial}{\partial t} + \vec{U} \cdot \nabla_{\vec{x}}$. Therefore, the transformed coordinates move with velocity $\vec{U}$.

Eq. 1 introduces 12 new variables, but they are not all independent. In order for the coordinate transformation to be single-valued, the coordinate differentials $d\xi_\alpha$ must be exact.[7] This is equivalent to requiring that each differential be expressible as the exterior derivative d of some scalar function f. For $d\xi$, this could be written:

$$d\xi = df = \frac{\partial f}{\partial \xi^\alpha} d\xi_\alpha \tag{14}$$

Similar equations hold for η and ζ. This implies that by differentiating df and using the requirement that mixed partial derivatives commute, any exact differential must satisfy the compatibility conditions:

$$\frac{\partial^2 x_\alpha}{\partial \xi^\alpha \partial \xi^\beta} = \frac{\partial^2 x_\alpha}{\partial \xi^\beta \partial \xi^\alpha} \Rightarrow \left\{ \begin{array}{lll} \frac{\partial A}{\partial \tau} = \frac{\partial U}{\partial \xi} & \frac{\partial B}{\partial \tau} = \frac{\partial V}{\partial \xi} & \frac{\partial C}{\partial \tau} = \frac{\partial W}{\partial \xi} \\ \frac{\partial L}{\partial \tau} = \frac{\partial U}{\partial \eta} & \frac{\partial M}{\partial \tau} = \frac{\partial V}{\partial \eta} & \frac{\partial N}{\partial \tau} = \frac{\partial W}{\partial \eta} \\ \frac{\partial P}{\partial \tau} = \frac{\partial U}{\partial \zeta} & \frac{\partial Q}{\partial \tau} = \frac{\partial V}{\partial \zeta} & \frac{\partial R}{\partial \tau} = \frac{\partial W}{\partial \zeta} \\ & & \\ \frac{\partial A}{\partial \eta} = \frac{\partial L}{\partial \xi} & \frac{\partial B}{\partial \eta} = \frac{\partial M}{\partial \xi} & \frac{\partial C}{\partial \eta} = \frac{\partial N}{\partial \xi} \\ \frac{\partial A}{\partial \zeta} = \frac{\partial P}{\partial \xi} & \frac{\partial B}{\partial \zeta} = \frac{\partial Q}{\partial \xi} & \frac{\partial C}{\partial \zeta} = \frac{\partial R}{\partial \xi} \\ \frac{\partial L}{\partial \zeta} = \frac{\partial P}{\partial \eta} & \frac{\partial M}{\partial \zeta} = \frac{\partial Q}{\partial \eta} & \frac{\partial N}{\partial \zeta} = \frac{\partial R}{\partial \eta} \end{array} \right. \tag{15}$$

It is important to note that these compatibility conditions are not independent either. The first nine equations are dependent on the temporal coordinate τ, and are sufficient to maintain the remaining conditions, provided that they are satisfied initially.

Provided that the constraints in Eq. 15 are satisfied, there is substantial freedom in choosing the grid velocity components U, V, W. This freedom can be exploited in order to yield coordinate systems that are better suited to resolving features specific to fluid dynamics, and to greatly simplify the grid-generation process.

One possible method of determining grid velocity is to require that η and ζ be material coordinates, such that:

$$\frac{D_{\vec{u}}\eta}{Dt} = \frac{\partial \eta}{\partial t} + \vec{u} \cdot \nabla_{\vec{x}} \eta = 0 \tag{16}$$

$$\frac{D_{\vec{u}}\zeta}{Dt} = \frac{\partial \zeta}{\partial t} + \vec{u} \cdot \nabla_{\vec{x}} \zeta = 0 \tag{17}$$

where $\vec{u}$ is the fluid velocity.

Eq. 16 imposes the requirement that the η coordinate of a fluid particle be "conserved". That is, the η- and ζ-coordinates of a fluid particle remain constant in time, effectively bounding fluid streamtubes with η and ζ coordinate surfaces. As a result, unphysical smearing at slip lines is largely eliminated, and a streamline-oriented mesh can be generated in a straightforward manner.

Combined with Eq. 13, these requirements yield the following formulas:

$$\left(\vec{u}-\vec{U}\right)\cdot\nabla_{\vec{x}}\eta = 0 \tag{18}$$

$$\left(\vec{u}-\vec{U}\right)\cdot\nabla_{\vec{x}}\zeta = 0 \tag{19}$$

In two dimensions, where $P = Q = C = N = \frac{\partial}{\partial\zeta} = 0$, $R = 1$, these reduce to:

$$\left(v - V\right)A = \left(u - U\right)B \tag{20}$$

$$W = 0 \tag{21}$$

One additional constraint may be imposed to determine the streamwise grid velocity component U, and several options are available, each useful under different circumstances. In two-dimensional flows, it is desirable to have an orthogonal grid, which can be obtained by constraining U to preserve grid angles, as follows [8]:

$$\frac{\partial}{\partial\tau}\cos^{-1}\left[\frac{\nabla_{\vec{x}}\xi\cdot\nabla\eta}{\left|\nabla_{\vec{x}}\xi\right|\left|\nabla\eta\right|}\right] = \frac{\partial}{\partial\tau}\cos^{-1}\left[\frac{AL+BM}{\sqrt{A^2+B^2}\sqrt{L^2+M^2}}\right] = 0 \tag{22}$$

After combining Eq. 22 with Eq. 20, we obtain(see Appendix A):

$$0 = \frac{\partial U}{\partial\eta} + \frac{\left|\nabla_{\vec{x}}\xi\right|^2 A}{\left|\nabla_{\vec{x}}\eta\right|^2 J}\left(A\frac{\partial v}{\partial\xi} - B\frac{\partial u}{\partial\xi}\right) - \frac{L}{J}\left(A\frac{\partial v}{\partial\eta} - B\frac{\partial u}{\partial\eta}\right) \tag{23}$$

$$+\left(\frac{\left|\nabla_{\vec{x}}\xi\right|^2}{\left|\nabla_{\vec{x}}\eta\right|^2 J}\left(\frac{\partial B}{\partial\xi}A + B\frac{\partial A}{\partial\xi}\right) - \frac{L}{JA}\left(\frac{\partial B}{\partial\eta}A + B\frac{\partial A}{\partial\eta}\right)\right)\left(U-u\right) \tag{24}$$

An orthogonal mesh is not possible to construct for three-dimensional flows, unless they belong to a class of flows called complex-lamellar, satisfying the equation:

$$\vec{v}\cdot\left(\nabla\times\vec{v}\right) = 0 \tag{25}$$

Therefore, it is not possible to preserve grid angles as a method of grid control in three dimensions. Alternative schemes include preserving grid skewness:

$$\kappa \equiv \frac{\left|\frac{\partial \vec{x}}{\partial \xi}\right| \cdot \left|\frac{\partial \vec{x}}{\partial \eta}\right| \cdot \left|\frac{\partial \vec{x}}{\partial \zeta}\right|}{\frac{\partial \vec{x}}{\partial \xi} \times \frac{\partial \vec{x}}{\partial \eta} \cdot \frac{\partial \vec{x}}{\partial \zeta}} - 1 \tag{26}$$

$$\frac{\partial \kappa}{\partial \tau} = 0 \tag{27}$$

or preserving the jacobian of the transformation:

$$\frac{\partial J}{\partial \tau} = 0 \tag{28}$$

It is possible to recover Lagrangian coordinates with the accompanying grid distortion by requiring $\frac{D_{\vec{u}}\xi}{Dt} = 0$.

Benefits & Challenges of Moving Coordinates By constraining the coordintes by as described above, it is possible to obtain an automatically generated, streamtube-oriented, low-distortion mesh. Doing so completely eliminates grid generation as part of the simulation process, and greatly improves solution accuracy at slip lines. The costs of this method are the increased computational costs from updating grid metric components, the cost of controlling grid distortion, and the computational costs of dealing with dynamic boundary conditions during the course of the simulation. As reported by Hui[7], the costs of updating the grid metric is negligible, and the cost of preserving grid angles is on the order of an extra 10% in simulation run-time.

Comparison with ALE Lagrangian-esque moving meshes invite comparison with Arbitrary-Lagrangian-Eulerian (ALE) schemes common in fluid-structure interaction research, but the two are in fact quite distinct. Although the present work does involve a grid that is, in a sense, a combination of Lagrangian and Eulerian grids, the implementation of ALE schemes is typically done very differently. In most ALE schemes, a fully Lagrangian grid is advanced in time for some arbitrary number of time steps before being interpolated onto an Eulerian grid to control distortion. This methodology is therefore unsuitable for grid generation, and the interpolation routine is a source of diffusion that can eliminate many of the accuracy advantages Lagrangian coordinates provide.

2.1.2 Euler Equations

Time-Dependent Euler Equations The unified coordinates were first published by Hui[7] in 1999, 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[9], 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[10].

2.1.3 Navier-Stokes and Recirculation

Hui and his team also successfully applied the unified coordinate system to the two-dimensional Navier-Stokes equations. Their principal applications were to boundary-layer flows. The first was a shock-boundary-layer interaction problem, composed of an incoming oblique shock impinging on a boundary layer to induce separation. They found that the unified coordinate system was quite capable of reproducing recirculating flow fields, despite its Lagrangian origins. The second problem was a shock-shock-interaction problem in a dual-ramp channel. They found that the unified coordinates produced results that were more accurate than a simulation using stationary, Eulerian coordinates, and did so with fewer grid points. Finally, they tested the Blasiussolution, in order to obtain a verification problem with which to quantify the accuracy of their method.

2.1.4 Other Equation Sets

Multi-material Flows Jia et al[11] applied Hui's method to multimaterial flows. They modeled immiscible multifluids with the interfaces described as contact discontinuities with good results.

MHD Flows Zhilkin[12] applied Hui's method to three-dimensional magnetohydrodynamics. Although only a preliminary analysis, he showed results for both decay of an MHD discontinuity and for an explosion in a magnetic field.

Kinetic Flows and BGK Analysis Jin and Xu[13][14] applied Hui's method to gas-kinetic BGK simulations of a freely falling flat plate.

Steady, Supersonic Euler Equations The space-marching equations were actually the first to be solved in the unified framework.

2.1.5 Summary

The unified coordinate system has been used and applied in a wide variety of fields. It possesses two primary advantages: first, automatic generation of a structured, body-fitted grid; second, streamline-alignment of the grid. Streamline-alignment leads to many useful properties, including excellent resolution of contact discontinuities.

3 Preliminary Work

Substantial work has already been done by the author at the Busemann Lab on the time-dependent Euler equations in two dimensions, including conference papers on a systematic verification study[15] and a preliminary attempt at boundary-layer integration and design work[16]. In short, the Busemann program can simulate simple, two-dimensional, inviscid flows without embedded surfaces and with a variety of boundary conditions.

3.1 Time-Dependent Euler Equations

In its original formulation, the time-dependent Euler equations were derived by Hui[7] with a grid velocity $\vec{U}$ given by

$$\vec{U} = h\vec{u} \tag{29}$$

where h is an arbitrary scalar function. Hui's development is summarized as follows.

$$\frac{\partial \mathbf{E}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial \xi} + \frac{\partial \mathbf{G}}{\partial \eta} = 0 \tag{30}$$

where

$$\mathbf{E} = \begin{pmatrix} \rho J \\ \rho J u \\ \rho J v \\ \rho J e \\ A \\ B \\ L \\ M \end{pmatrix} \tag{31}$$

$$\mathbf{F} = \begin{pmatrix} \rho J (1-h) u_{\vec{\xi}} \\ \rho J (1-h) u_{\vec{\xi}} u + pM \\ \rho J (1-h) u_{\vec{\xi}} v - pL \\ \rho J (1-h) u_{\vec{\xi}} e + p u_{\vec{\xi}} \\ -hu \\ -hv \\ 0 \\ 0 \end{pmatrix}, \mathbf{G} = \begin{pmatrix} \rho J (1-h) u_{\vec{\eta}} \\ \rho J (1-h) u u_{\vec{\eta}} - pB \\ \rho J (1-h) v u_{\vec{\eta}} + pA \\ \rho J (1-h) e u_{\vec{\eta}} + p u_{\vec{\eta}} \\ 0 \\ 0 \\ -hu \\ -hv \end{pmatrix} \tag{32}$$

and

$$J = AM - BL,\ u_{\vec{\xi}} = \frac{uM - vL}{J},\ u_{\vec{\eta}} = \frac{Av - Bu}{J} \tag{33}$$

where J represents the jacobian of the transformation, and is equivalent to the physical volume of each computational grid cell. $u_{\vec{\xi}}$ and $u_{\vec{\eta}}$ 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. 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

$$\alpha(\xi,\eta)\frac{\partial g}{\partial \xi} + \beta(\xi,\eta)\frac{\partial g}{\partial \eta} + \gamma(\xi,\eta) = 0 \tag{34}$$

where

$$\alpha = \left(L^2 + M^2\right)^2 (A\cos\theta - B\sin\theta) \tag{35}$$

$$\beta = \left(A^2 + B^2\right)^2 (M\cos\theta - L\sin\theta) \tag{36}$$

$$\gamma = -\left(L^2 + M^2\right)^2 (A\cos\theta + B\sin\theta)\frac{\partial\theta}{\partial\xi} \tag{37}$$

$$+\left(A^2 + B^2\right)^2 (L\cos\theta + M\sin\theta)\frac{\partial\theta}{\partial\eta} \tag{38}$$

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. Iterative relaxation schemes are also effective, even if the coefficients do change sign.

As Eq. 30 is in strong conservation form, it can be solved by established shock-capturing methods. We use the Godunov scheme with MUSCL refinement given by Hui[7]. 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 thecomputational 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[8].

3.1.1 Spatial Discretization

The spatial discretization scheme used 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 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. 34 can ensure that neighboring nodes remain neighbors throughout the simulation, thus obviating the need for connectivity changes.

3.1.2 Flow Solver

As mentioned above, Eq. 30 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[7] 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 three parts: the conservation equations for the primitive, physical variables p, ρ, u, v and w form the first group; the geometric conservation equations for A, B, C, L, M, N, P, Q, R form the second group; and the equations for grid velocity U, V, W form 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 $\mathcal{L}_{\xi\eta}^{\Delta\tau}$ is approximated as $\mathcal{L}_{\xi}^{\Delta\tau/2}\mathcal{L}_{\eta}^{\Delta\tau}\mathcal{L}_{\xi}^{\Delta\tau/2}$.

The full algorithm used at the Busemann Lab is given as follows (Fig. 1). For each time step:

1. Compute optimal time step[17]. 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. 34 for h and update coordinate positions using trapezoidal integration.

4. Add/remove columns of nodes as needed.

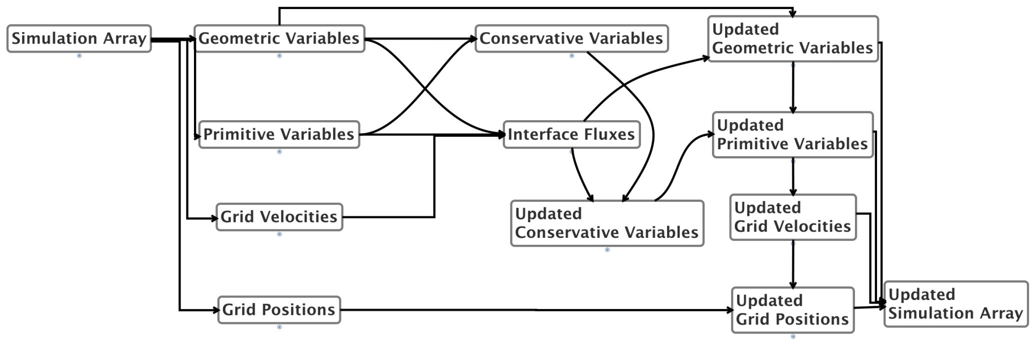


Figure 1: A diagram of information flow in a unified coordinates program

By artificially decoupling the equations that govern the evolution of the flow from those that govern the metric, the conservation equations (Eq. 30) 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 transformed equations by decoupling only the equation for h and making additional approximations[9], but the end resulting algorithm is identical in either case.

3.1.3 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.[18] The specific implementation given by Hui [7] is given as:

$$\begin{gathered}
f_r = f_{i+1,j} - 0.5\left(f_{i+2,j} - f_{i+1,j}\right)\phi\left(r^+\right) \\
r^+ = \frac{\left(f_{i+1,j} - f_{i,j}\right)}{\left(f_{i+2,j} - f_{i+1,j}\right)} \\
f_l = f_{i,j} + 0.5\left(f_{i,j} - f_{i-1,j}\right)\phi\left(r^-\right) \\
r^- = \frac{\left(f_{i+1,j} - f_{i,j}\right)}{\left(f_{i,j} - f_{i-1,j}\right)} \\
\phi\left(r\right) = \max\left(0, \min\left(1, r\right)\right)
\end{gathered} \tag{39}$$

3.1.4 Boundary Conditions

Boundary conditions are implemented through the use of ghost cells as seen in Fig. 2, 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.

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.

Grid Communication Boundary In order to introduce flows consisting of more than a single streamtube, such as would be required for a

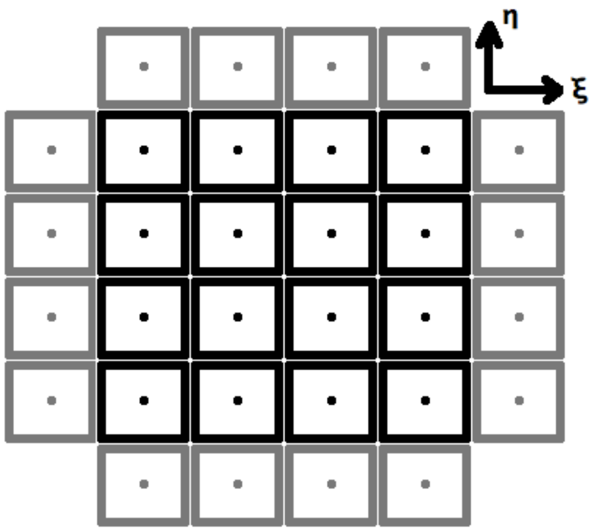


Figure 2: Diagram of computational ξ-η space. Ghost cells are shown in gray.

Table 1: Application of boundary conditions

Boundary Type	**Specified Quantities**	**How Enforced**
Supersonic Inflow	All	Entire flow state is directly specified
Supersonic Outflow	None	Flow state is copied from simulation interior
Slip (Inviscid 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.

two-dimensional airfoil, it is necessary to implement some form of communication between separate grids. One way to do this is via boundary conditions applied at grid interfaces. This will be done with ghost cells as well.

3.1.5 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.

$$x_{i,j}^{n+1} = x_{i,j}^{n} + \frac{1}{2}\left(U_{i,j}^{n} + U_{i,j}^{n+1}\right)\Delta\tau \tag{40}$$

$$y_{i,j}^{n+1} = y_{i,j}^{n} + \frac{1}{2}\left(V_{i,j}^{n} + V_{i,j}^{n+1}\right)\Delta\tau z_{i,j}^{n+1} = \quad z_{i,j}^{n} + \frac{1}{2}\left(W_{i,j}^{n} + W_{i,j}^{n+1}\right)\Delta\tau \tag{41}$$

3.2 Steady Riemann Problem

In order to perform a reliable verification, it is important to stress the solver. The Riemann problem given by Hui[7], with its three distinct nonlinear waves, provides such a stress. It is given by the upstream boundary condition (See Fig. 3):

$$(p, \rho, Ma, \theta) = \begin{cases} (0.25, 0.5, 7, 0) & : y > 0 \\ (1, 1, 2.4, 0) & : y < 0 \end{cases} \tag{42}$$

where Ma is the Mach number and θ is the flow angle given by $u = |\vec{u}|\cos\theta$, $v = |\vec{u}|\sin\theta$.

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. 4(a) and 4(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.

Fig. 5 quantifies these findings. The solution is shown to converge in three different test configurations, albeit with lower convergence rates than were expected. However, some recent results have been obtained for other verification problems, and the end behavior is not yet understood. Further work is required in this area.

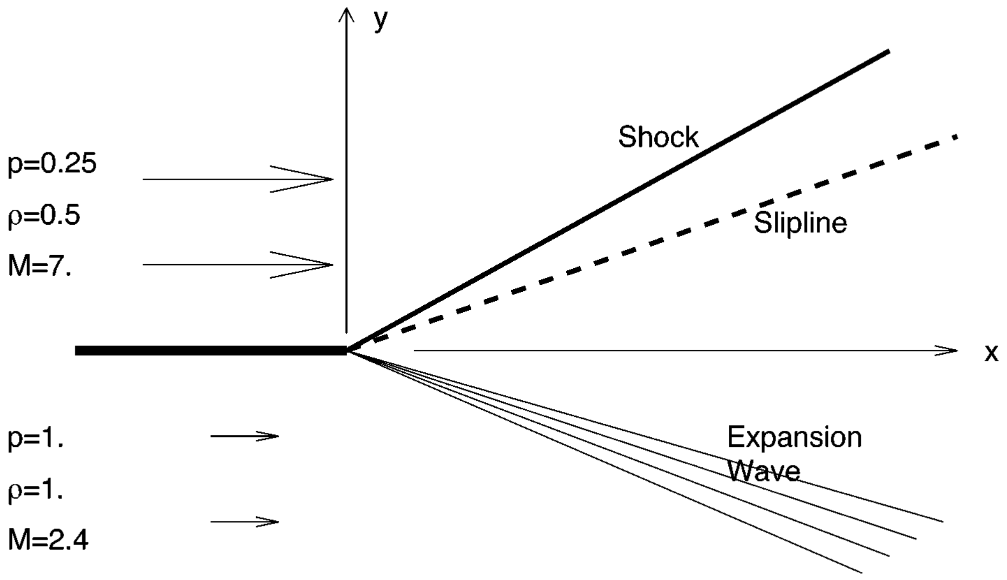
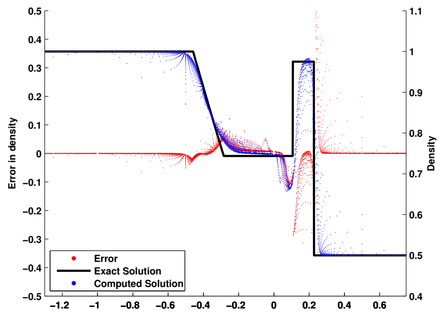
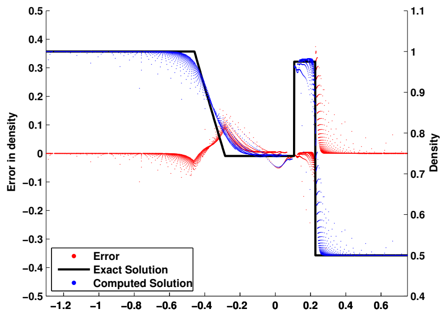


Figure 3: A diagram of the steady Riemann problem, as given by Hui[7]



(a) $h = 0$ (Eulerian)



(b) $h = 0.999$ (Unified)

Figure 4: The similarity solution of the Riemann problem and the corresponding error in the numerical solution, computed throughout the simulation region.

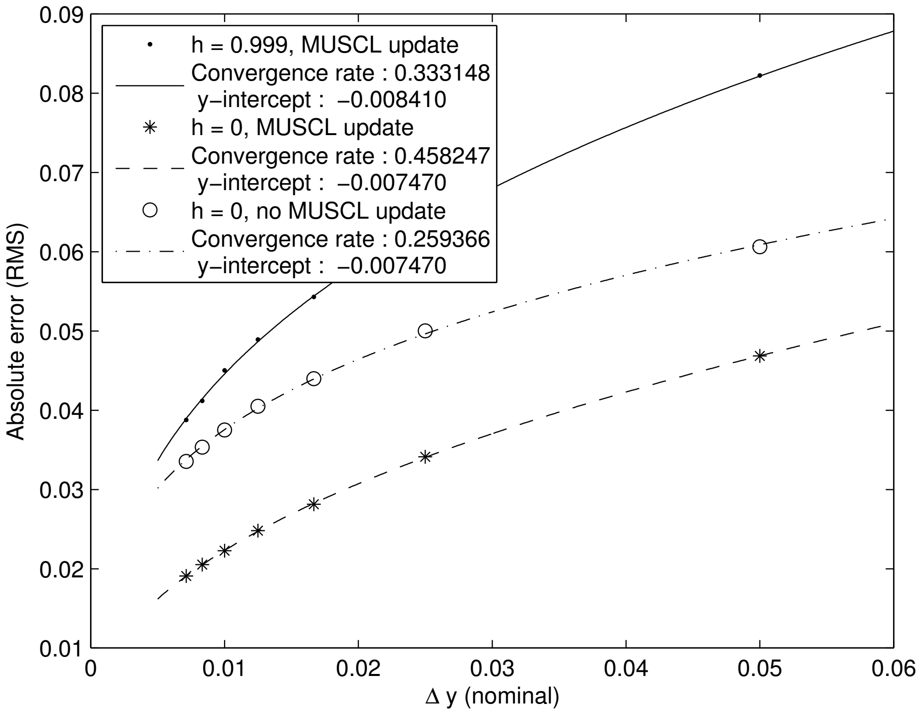


Figure 5: Root-mean-squared error for the Riemann problem, with order of convergence n.

3.3 The Oblique Shock

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

$$(p, \rho, M, \theta) = (1, 1, 1.8, 0) \tag{43}$$

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 preliminary grid convergence study was performed, and the results are shown in Fig. 6.

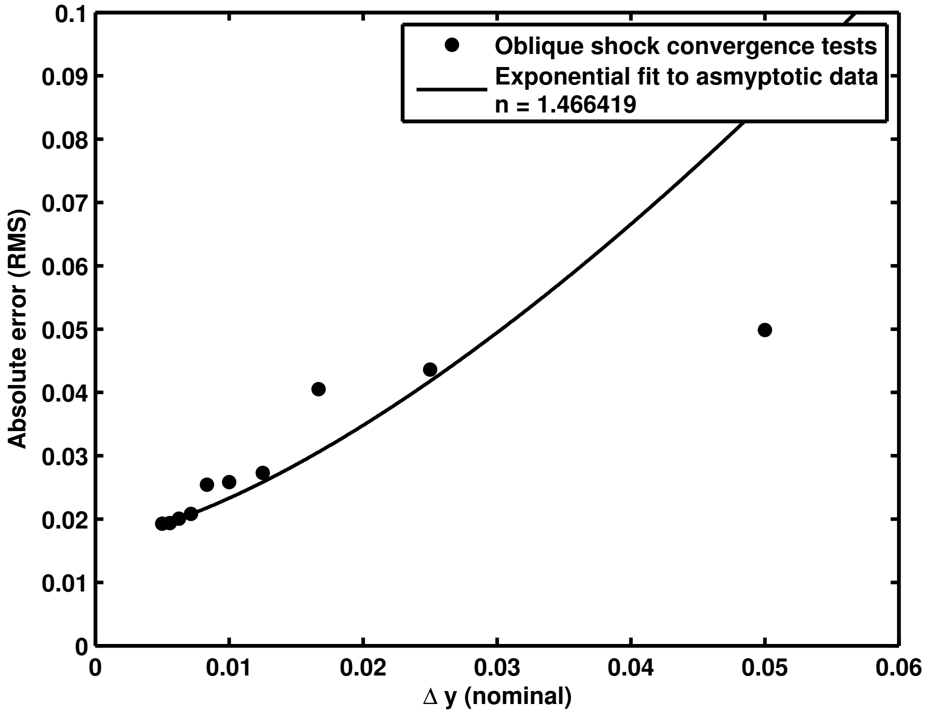


Figure 6: Root-mean-squared error for the oblique shock problem. Curve is fit to the four highest resolution points only.

3.4 Prandtl-Meyer Expansion

3.4.1 The Expansion Fan and "Grid Separation"

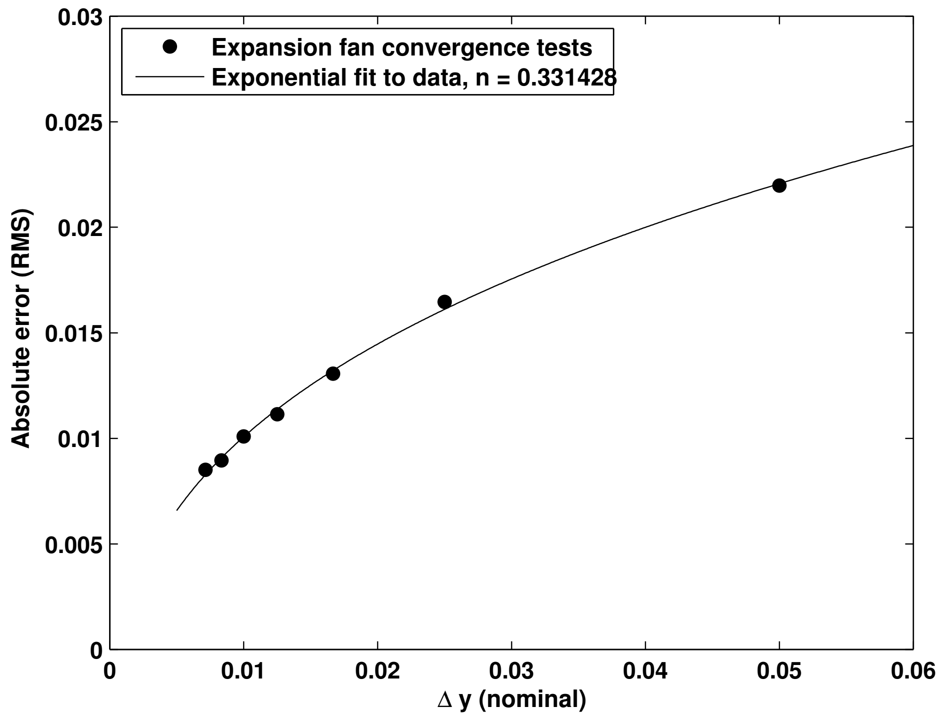


Figure 7: Root-mean-squared error for the Prandtl-Meyer expansion, with order of convergence n.

A final verification test was run for a Prandtl-Meyer expansion corner. Convergence was again demonstrated (see Fig. 7), but the expansion corner reveals an important limitation of the current 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. 8. Thus, although the streamlines eventually do align with the wall as expected, the dramatic loss of resolution near the wall may seriously affect any further downstream results.

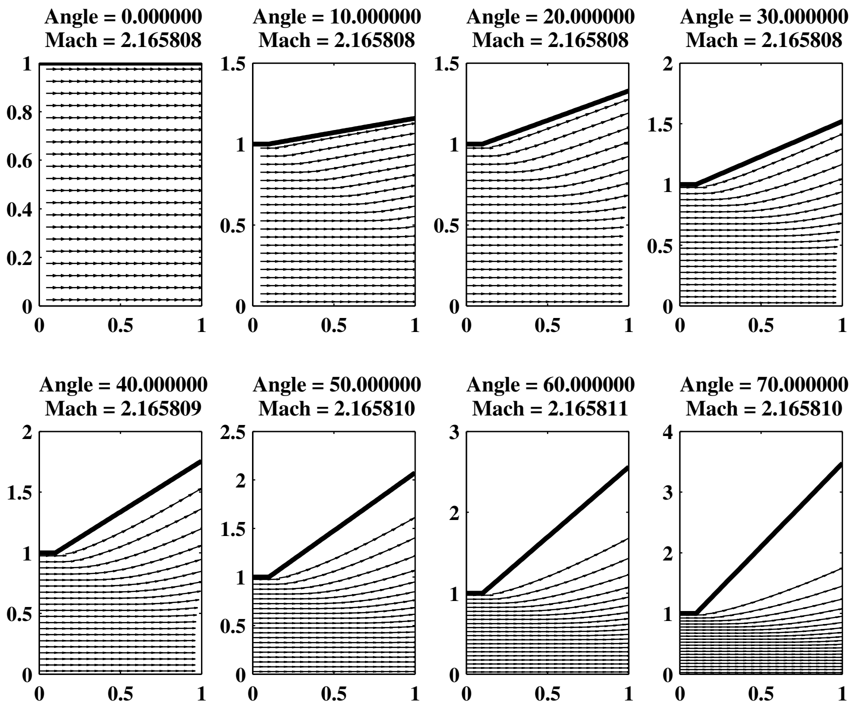


Figure 8: Computed streamlines for expansion at increasing angles. Angles are given in degrees.

3.5 Diamond Shock Train

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. 9, 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. In contrast, using traditional methods, 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.

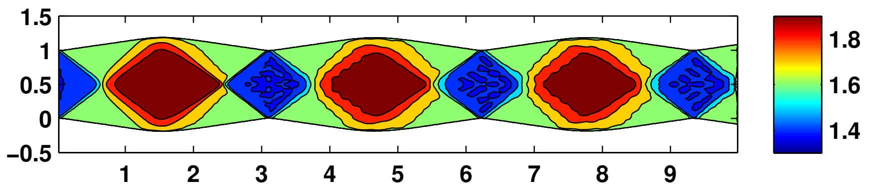


Figure 9: Computed Mach number for an under-expanded nozzle flow, showing the diamond-shock train

3.6 A Transonic Duct Flow

Finally, we use the unified coordinates method to model flow through a convergent duct. This problem is identical to the one given by Hui[7], and is characterized by the formation of a mach stem and the resulting transonic flow (Fig. 10). The mesh again flows through the duct, conforming to solid walls without need for user specification of the mesh.

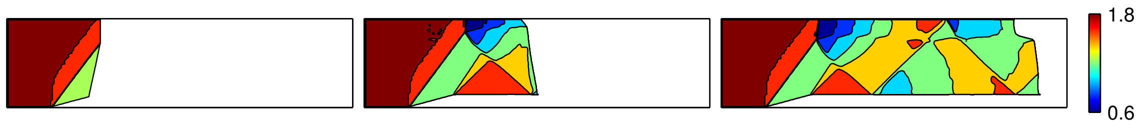
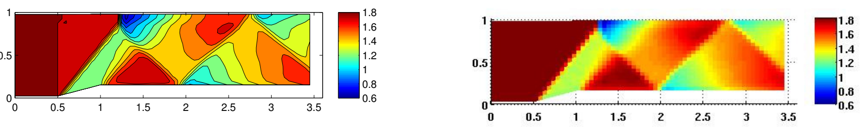
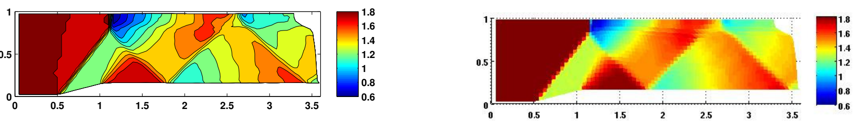


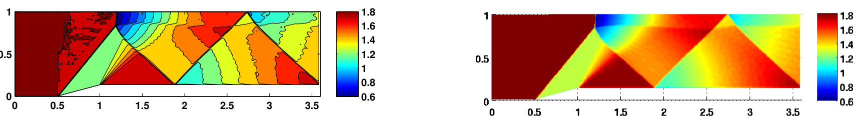
Figure 10: Computed Mach number for transonic duct flow at various times, showing the manner in which nodes flow to fill boundaries.



(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 11: Qualitative accuracy comparison between Unified ($h = 0.25$) (b) and Eulerian ($h = 0$) (a,c) simulations for a transonic duct flow. Notice the improved resolution of the slip line and the walls for the unified solution.

It can be seen in Fig. 11 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 unified coordinates 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.

3.7 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. As a result, some method of accounting for viscous effects is almost a requirement for hypersonic flows, and boundary-layer methods provide this at minimal cost.

A crude, prototypical implementation uses the turbulent, flat-plate, constant pressure formula given in Schlichting[19]:

$$\frac{\delta u_\infty}{\nu} = 0.14 \frac{Re_x}{\log Re_x} G(\log Re_x) \tag{44}$$

where G is taken to be the limiting value of 1.

This serves as a useful proof-of-concept for the method, and will be a valuable step toward implementation of an actual boundary-layer solver. In the inviscid simulation, boundary-layer 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. 12. 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.

3.8 Model F-14 Inlet

One of the principal attractions of the unified coordinates 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

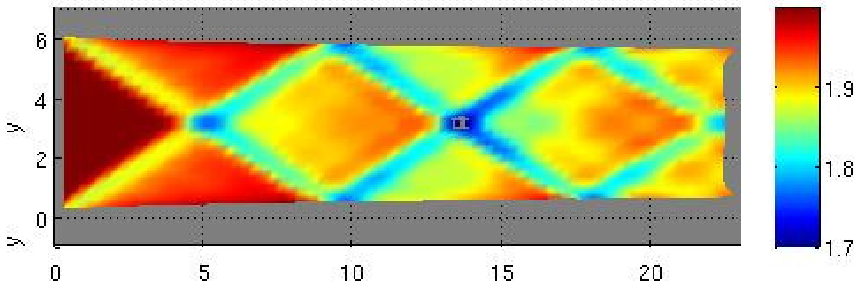


Figure 12: Oblique shock train produced by a turbulent boundary layer in an otherwise uniform channel

number in the range $0 < M < 2.3$. This is accomplished through the use of internal, variable ramps, as shown in Fig. 13.

This is exactly the kind of problem the unified coordinates 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.

An example based on the F-14 inlet is shown in Fig. 14, but the results are only prototypical. Due to as-yet-unimplemented features such as multi-stream capability and subsonic outflow conditions, it is not possible to correctly model spillage, nor is it possible to achieve subsonic flow in the inlet. Further testing will be possible once these features have been added.

4 Proposal & Timeline

Although much has already been done to advance the capabilities of the unified coordinate system, more remains if it is to become a successful design tool. The disparate pieces discussed in Sec. 2 need to be combined in an intuitive, automatic way, that minimizes human interaction and keeps computational costs low. I propose to create this design tool by combining inviscid, viscous, and boundary-layer solvers within a wrapping program that handles grid generation, boundary communication, and solver selection. The wrapper will do this automatically, without the need for human interaction, and requiring only the boundary conditions and fluid properties as inputs.

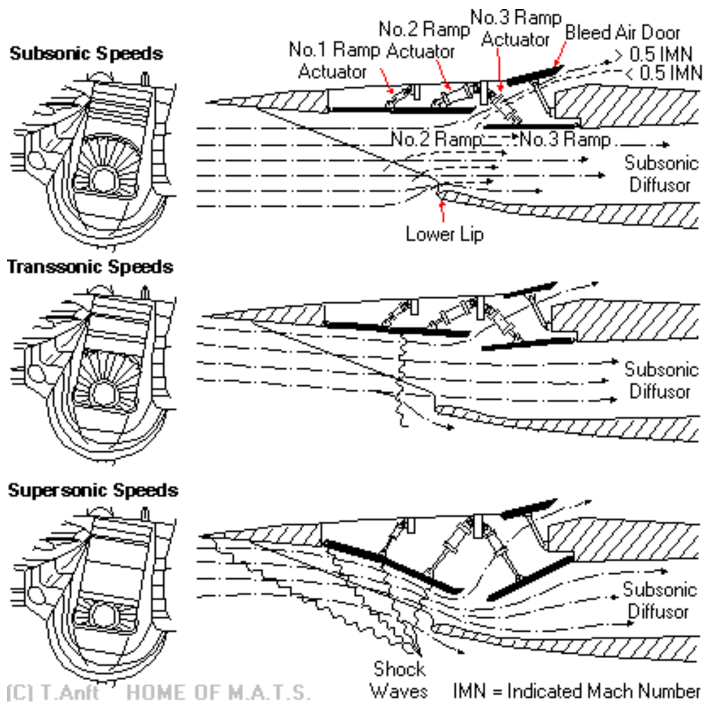


Figure 13: Diagram of the variable inlet geometry of the USAF F-14 Tomcat. Courtesy of: Home of M.A.T.S., Available at http://www.anft.net/f-14/f14-detail-airintake.htm, Accessed 25 Nov 2011

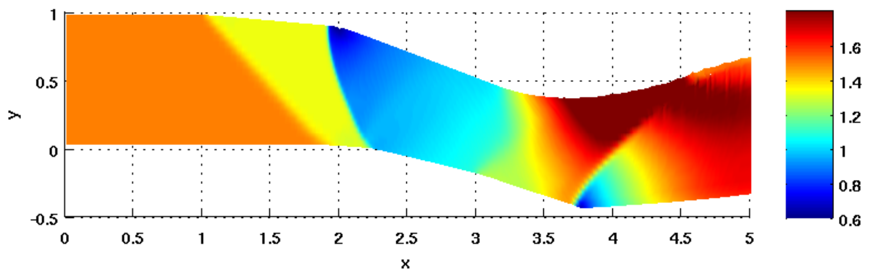


Figure 14: Mach number in a geometry modeled after Fig. 13.

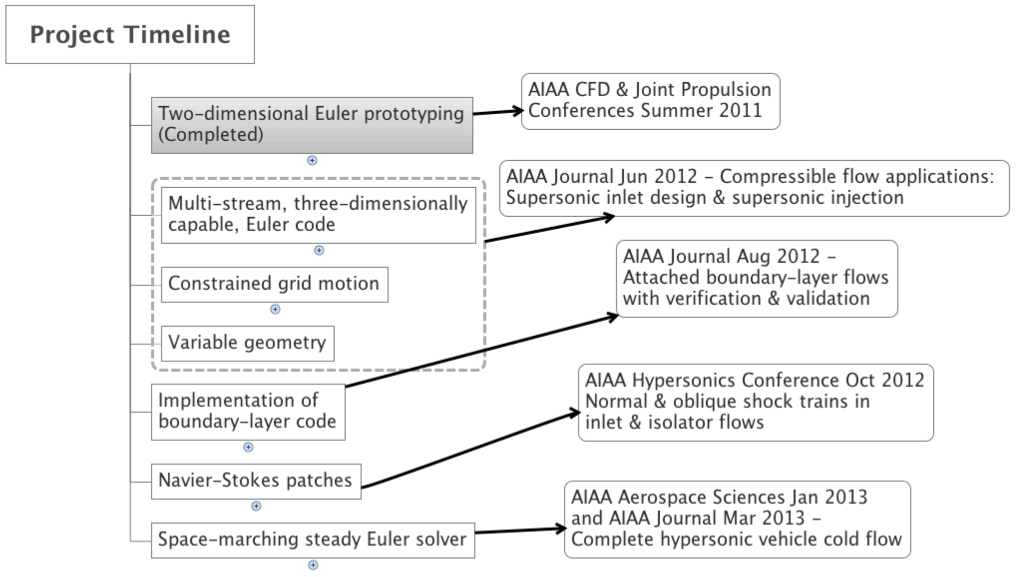

Project Timeline
Two-dimensional Euler prototyping (Completed)
AIAA CFD & Joint Propulsion Conferences Summer 2011
Multi-stream, three-dimensionally capable, Euler code
Constrained grid motion
Variable geometry
AIAA Journal Jun 2012 – Compressible flow applications: Supersonic inlet design & supersonic injection
Implementation of boundary-layer code
AIAA Journal Aug 2012 – Attached boundary-layer flows with verification & validation
Navier-Stokes patches
AIAA Hypersonics Conference Oct 2012 Normal & oblique shock trains in inlet & isolator flows
Space-marching steady Euler solver
AIAA Aerospace Sciences Jan 2013 and AIAA Journal Mar 2013 – Complete hypersonic vehicle cold flow

Figure 15: Proposed timeline for dissertation work & publications

A specific project timeline with expected publications is shown in Fig. 15.

4.1 Multi-Stream, Three-Dimensionally Capable Inviscid Solver

Although a working time-marching code is available already for two-dimensional problems, a new code will be written to include also additional capability and features, such as extensibility to multi-stream flows and to three-dimensional flows. Although 3-D brings with it a host of new problems for which solutions are not yet known, the basic changes to the equation sets are actually quite simple. Therefore, the code will be written to handle the three-dimensional equations, but the methods for handling three-dimensional boundary conditions will not be developed.

4.1.1 Three-Dimensional Flows

Aside - Why not a fully 3-D code Three-dimensional flows are much, much more complex than two-dimensional flows are, no matter what the solver being used. Unfortunately, because of the unsteady nature of grid cell boundary conditions in moving grid systems, it would be difficult to directly leverage the work that has been done in traditional CFD to solve these problems, and resolving them would therefore require a substantial investment of time and effort which could be better spent proving the concepts in the simpler two-dimensional case, while still enabling many useful applications.

For methods relying on moving grid systems, the most obvious difficulty is the way in which complex geometries are handled. The marching grid technique discussed in Sec. 3.1.1must automatically detect downstream flow boundaries and deal with complex, transient, streamtube bending and tangling. The dynamic update of boundary conditions is also more difficult. These issues exist for all complex flows, but the two-dimensional case is much simpler.

4.1.2 Multi-Stream Flows

The work that has been done at the Busemann Advanced Concepts Lab on moving mesh CFD has thus far included only simple flows composed of a single streamtube. Since that excludes such classic problems as flow around airfoils, multi-stream capability is an essential component for any useful design program.

Separate Grids Rather than attempt to work with complex and variable topology on a single grid, multiple streams will be handled by multiple separate grids which communicate with one another through their boundary conditions. For the second-order MUSCL scheme used here, this mandates the use of a double-layer of ghost cells between grids.

Aside - MPI Parallelism It is worth noting that a multi-stream approach is directly adaptable to distributed-memory parallelism. This will greatly simplify the eventual parallelization of the program.

4.1.3 Goals

- Communication between multiple streams of inviscid flow - Completed by: Mar, 2012, Paper on variable geometry supersonic inlet submitted to AIAA Journal, 30 Apr
- Sample problems - Supersonic injection, wing cross-section with verification and validation, supersonic inlet (F-14, Concorde)

4.2 Constraining Grid Motion

Thus far, the Busemann Lab has not implemented a robust method for grid distortion control. Doing so is a light project, and will require the derivation of control equations for jacobian- and skewness-preserving schemes as well for grid-angle-preserving schemes. It will also require the development and implementation of numerical grid solvers.

4.2.1 Goals

Second-order method for grid-angle-preserving implemented - Completed by Mar 2012 ODEs derived for jacobian-preserving and skewness-preserving - Completed by Apr 2012

4.3 Boundary-Layer Implementation

The Busemann Advanced Concepts Lab has access to a legacy boundary-layer code from NASA, which will be integrated with an inviscid core solver for basic viscous modeling. The integration is done as follows: a new boundary will be defined that approximates the calculated displacement thickness of the boundary-layer, and the process will be iterated until convergence.

4.3.1 Goals

- Communication between a boundary-layer solver and the inviscid core - Completed by Jun 2011, Paper on verification & validation submitted to AIAA Journal by Aug 2012

- Sample problems - Flat plate boundary layer, NACA airfoils

4.4 Navier-Stokes Solver

Although it has been demonstrated that the full Navier-Stokes equations can be solved throughout the simulation region, the computational cost of doing so would be unnecessarily high. An alternative is to solve the Navier-Stokes equations only where simpler methods are insufficient, such as in regions of separated flow. This builds heavily on the multi-stream capability that will have been developed previously.

4.4.1 Stationary Grid Patches and Material Coordinates

Since the primary regions of interest for Navier-Stokes solutions are regions of recirculation, the use of material coordinates may not be appropriate. Hui found that recirculation was quite easily handled using a variation of the grid-angle-preserving technique, and that it was also possible to pin the grid in place, a technique that will be especially useful for resolving recirculation bubbles.[17]

4.4.2 Boundary Conditions on Viscous Patches

The boundary conditions for viscous flow patches can be implemented much as for any other patch. Patches can be constrained so that grid points align, but without such a constraint it will be necessary to account for non-aligned grid points at the boundary.

4.4.3 Relationship With Boundary-Layer Solver

Navier-Stokes patches could, in principle, be used to compute any viscous flow, but they are most efficiently used in regions where boundary-layer solvers are no longer valid. As a result, the patches will be used only after the separation point, which is known from the boundary-layer solver.

4.4.4 Goals

- Manual creation of Navier-Stokes patches for the solution of separation bubbles - Completed by: Jun 2012, Normal & oblique shock trains for isolators presented at AIAA Hypersonics conference, Oct. 2012
- Automation of Navier-Stokes bubbles - Completed by: Sept 2012 Presented at AIAA Aerospace Sciences conference, Jan 2013
- Sample problems - Isolator shock trains, Stalled wings, Separated inlet flows

4.5 Space-Marching Euler Solver

Space-marching has long been known as an extremely efficient alternative to time-marching for steady, supersonic flows, reducing computational costs by several orders of magnitude. This makes space-marching extremely attractive for supersonic design work, provided that it can be combined with more general methods to handle subsonic and time-dependent flows.

Dealing with Bow Shocks and other Normal Shocks It is possible to accurately predict both bow shock shape and standoff distance quite accurately for hypersonic, blunt-body flows. Anderson reports an empirical fit which gives the following formulas for shock shape x and standoff distance δ[20]:

$$x = R_c + \delta - R_s \cot^2 \beta \left[\sqrt{1 + \frac{y^2 \tan^2 \beta}{R_s^2}} - 1 \right] \tag{45}$$

$$\frac{\delta}{R_c} = \begin{cases} 0.143 \exp\left[3.24/Ma_\infty^2\right] & \text{sphere} - \text{cone} \\ 0.386 \exp\left[4.67/Ma_\infty^2\right] & \text{cylinder} - \text{wedge} \end{cases} \tag{46}$$

$$\frac{R_s}{R_c} = \begin{cases} 1.143 \exp\left[0.54/\left(Ma_\infty - 1\right)^{1.2}\right] & \text{sphere} - \text{cone} \\ 1.386 \exp\left[1.8/\left(Ma_\infty - 1\right)^{0.75}\right] & \text{cylinder} - \text{wedge} \end{cases} \tag{47}$$

In the above, R_c is the radius of curvature of the blunt body, R_s is the radius of curvature of the bow shock, δ is the standoff distance $R_c - R_s$, and β is either the angle of the attached shock corresponding to the body shape further downstream (i.e. wedge, cone, etc.) if had a sharp leading edge, or the angle of a Mach wave in the case of a downstream body that is parallel to the incoming flow. The sphere-cone relations correspond to axisymmetric flows, while the cylinder-wedge relations correspond to two-dimensional flows. Hence, for this project, it is possible to compute standoff distance δ and start a time-dependent patch upstream of that. The time-dependent patch will be run such that it encompasses the subsonic region and the nearby supersonic flow until it reaches a steady-state. Space-marching can then continue around the subsonic bubble as before.

An alternative methodology was used by Rizzi[21] and detailed in Anderson[22], which uses a grid that moves with the bow shock speed, thus creating a shock-fitted solution to the subsonic region. This may prove to be an attractive alternative to computing a standoff distance, as the time-dependent equations will only be used where necessary, and the shock-fitted solution may be more accurate.

4.5.1 Goals

- Completed by: Dec 2013, Presented at AIAA Aerospace Sciences conference, Jan 2013
- Sample problems - Shock-shock interaction, 2-D hypersonic vehicle (X-51, X-43), Re-entry vehicle

References

[1] A Jameson. Re-engineering the design process through computation. *Journal of Aircraft*, 36:36–50, January 1997.

[2] E. Steinthorsson and D. Modiano. Advanced methodology for simulation of complex flows using structured grid systems. In *Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamic (CFD) Solutions p 697-710 (SEE N95-28723 10-02)*, pages 697–710, March 1995.

[3] F.T. Johnson, E.N. Tinoco, and N.J. Yu. Thirty years of development and application of cfd at boeing commercial airplanes, seattle. *Computers & Fluids*, 34(10):1115 – 1151, 2005.

[4] K.J. Badcock, B.E. Richards, and M.A. Woodgate. Elements of computational fluid dynamics on block structured grids using implicit solvers. *Progress in Aerospace Sciences*, 36(5-6):351 – 392, 2000.

[5] D. Garretson, H. Mair, C. Martin, K. Sullivan, and J. Teichman. Review of cfd capabilities. Technical report, Institute for Defense Analyses, Science & Technology Division, Alexandria, VA, September 2005.

[6] V. Venkatakrishnan. A perspective on unstructured grid flow solvers. Technical report, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, February 1995.

[7] W. H. Hui, P. Y. Li, and Z. W. Li. A unified coordinate system for solving the two-dimensional euler equations. *Journal of Computational Physics*, 153:596–673, 1999.

[8] W. H. Hui. The unified coordinate system in computational fluid dynamics. *Communications in Computational Physics*, 2(4):577–610, August 2007.

[9] W. H. Hui and S. Kudriakov. A unified coordinate system for solving the three-dimensional euler equations. *Journal of Computational Physics*, 172:235–260, January 2001.

[10] W. Hui, J. Hu, and G. Zhao. Gridless computation using the unified coordinates. In Clinton Groth and David W. Zingg, editors, *Computational Fluid Dynamics 2004*, pages 503–508. Springer Berlin Heidelberg, 2006.

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

[12] A.G. Zhilkin. A dynamic mesh adaptation method for magnetohydrodynamics problems. *Computational Mathematics and Mathematical Physics*, 47(11):1819–1832, 2007.

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

[14] C. Jin and K. Xu. Numerical study of the unsteady aerodynamics of freely falling plates. *Communications in Computational Physics*, 3(4):834–851, April 2008.

[15] C. N. Woods and R. P. Starkey. Verification and application of the unified coordinate system in preliminary design. In *Proceedings of the 20th AIAA Computational Fluid Dynamics Conference*, Honolulu, HI, June 2011.

[16] C. N. Woods and R. P. Starkey. Propulsion flowpath design using an automatic-mesh, eulerian/lagrangian cfd approach. In *Proceedings of the 47th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, San Diego, CA, July 2011.

[17] W.H. Hui, Z. N. Wu, and B. Gao. Preliminary extension of the unified coordinate system approach to computation of viscous flows. *Journal of Scientific Computing*, 30(2):301–344, February 2007.

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

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

[20] J. D. Anderson. *Hypersonic and High-Temperature Gas Dynamics.* AIAA Education Series. American Institute of Aeronautics and Astronautics, 2nd edition, 2006.

[21] A. Rizzi and H. Bailey. Finite-volume solution of the euler equations for steady three-dimensional transonic flow. In Adriaan van de Vooren and Pieter Zandbergen, editors, *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 July 2, 1976 Twente University, Enschede*, volume 59 of *Lecture Notes in Physics*, pages 347–352. Springer Berlin / Heidelberg, 1976.

[22] J.D. Anderson. *Modern compressible flow: with historical perspective.* McGraw-Hill series in aeronautical and aerospace engineering. McGraw-Hill, 2004.

A Derivation of Grid-Angle-Preserving ODE for U

The requirement that grid angles be preserved can be written:

$$\frac{\partial}{\partial \tau}\left[\cos^{-1}\left(\frac{\nabla\xi\cdot\nabla\eta}{|\nabla\xi|\,|\nabla\eta|}\right)\right] = \frac{\partial}{\partial \tau}\left[\cos^{-1}\left(\frac{AL+BM}{\sqrt{A^2+B^2}\sqrt{L^2+M^2}}\right)\right] = 0 \quad (48)$$

We define

$$S^2 \equiv L^2 + M^2 \quad (49)$$

$$T^2 \equiv A^2 + B^2 \quad (50)$$

$$J \equiv AM - BL \quad (51)$$

to rewrite Eq. 48 as

$$\frac{\partial}{\partial \tau}(AL+BM)-(AL+BM)\left(\frac{A\dot{A}+B\dot{B}}{T^2}+\frac{L\dot{L}+M\dot{M}}{S^2}\right)=0 \tag{52}$$

$$\begin{aligned}\Rightarrow 0 &= \left(L-\frac{(AL+BM)}{T^2}A\right)\dot{A}+\left(M-\frac{(AL+BM)}{T^2}B\right)\dot{B}\\ &+\left(A-\frac{(AL+BM)}{T^2}L\right)\dot{L}+\left(B-\frac{(AL+BM)}{T^2}M\right)\dot{M}\end{aligned} \tag{53}$$

$$\Rightarrow 0=-\frac{JB}{T^2}\dot{A}+\frac{JA}{T^2}\dot{B}+\frac{JM}{S^2}\dot{L}-\frac{JL}{S^2}\dot{M} \tag{54}$$

From the requirement that $\frac{D_{\vec{u}}\eta}{Dt}=0$ and the compatibility conditions, we have:

$$V=v-\frac{B}{A}(u-U) \tag{55}$$

$$\Rightarrow V_x=v_x+\frac{B_xA+BA_x}{A^2}(U-u)+\frac{B}{A}(U_x-u_x) \tag{56}$$

$$\begin{array}{ll}\dot{A}=U_\xi & \dot{B}=V_\xi\\ \dot{L}=U_\eta & \dot{M}=V_\eta\end{array} \tag{57}$$

Substituting into our equation, we have

$$\Rightarrow 0=-\frac{JB}{T^2}U_\xi+\frac{JA}{T^2}V_\xi+\frac{JM}{S^2}U_\eta-\frac{JL}{S^2}V_\eta \tag{58}$$

$$\begin{aligned}\Rightarrow 0 &= -\frac{JB}{T^2}U_\xi+\frac{JA}{T^2}\left(v_\xi+\frac{B_\xi A+BA_\xi}{A^2}(U-u)+\frac{B}{A}(U_\xi-u_\xi)\right)\\ &+\frac{JM}{S^2}U_\eta-\frac{JL}{S^2}\left(v_\eta+\frac{B_\eta A+BA_\eta}{A^2}(U-u)+\frac{B}{A}(U_\eta-u_\eta)\right)\end{aligned} \tag{59}$$

We separate the equation into terms of U, U_ξ, U_η:

$$0=\left(-\frac{JB}{T^2}+\frac{JA}{T^2}\frac{B}{A}\right)U_\xi+\left(\frac{JM}{S^2}-\frac{JL}{S^2}\frac{B}{A}\right)U_\eta \tag{60}$$

$$+\left(\frac{JA}{T^2A^2}(B_\xi A+BA_\xi)-\frac{JL}{S^2A^2}(B_\eta A+BA_\eta)\right)(U-u) \tag{61}$$

$$+\frac{JA}{T^2}\left(v_\xi-\frac{B}{A}u_\xi\right)-\frac{JL}{S^2}\left(v_\eta-\frac{B}{A}u_\eta\right) \tag{62}$$

$$\Rightarrow 0 = U_\eta + \frac{S^2 A}{T^2 J}\left(Av_\xi - Bu_\xi\right) - \frac{L}{J}\left(Av_\eta - Bu_\eta\right) \tag{63}$$

$$+\left(\frac{S^2}{T^2 J}\left(B_\xi A + BA_\xi\right) - \frac{L}{JA}\left(B_\eta A + BA_\eta\right)\right)(U - u) \tag{64}$$