A GENERALIZED FLUID FORMULATION FOR TURBOMACHINERY COMPUTATIONS

Charles L. Merkle*, Venkateswaran Sankaran*,
Daniel J. Dorney#, and Douglas L. Sondak+

*University of Tennessee Space Institute,
#NASA Marshall Space Flight Center and
+Boston University

AIAA Paper 2003-3999
33rd AIAA Fluid Dynamics Conference and Exhibit
23-26 June 2003
Hilton at Walt Disney World
Orlando, Florida

Abstract

A generalized formulation of the equations of motion of an arbitrary fluid are developed for the purpose of defining a common iterative algorithm for computational procedures. The method makes use of the equations of motion in conservation form with separate pseudo-time derivatives used for defining the numerical flux for a Riemann solver and the convergence algorithm. The partial differential equations are complemented by an thermodynamic and caloric equations of state of a complexity necessary for describing the fluid. Representative solutions with a new code based on this general equation formulation are provided for three turbomachinery problems. The first uses air as a working fluid while the second uses gaseous oxygen in a regime in which real gas effects are of little importance. These nearly perfect gas computations provide a basis for comparing with existing perfect gas code computations. The third case is for the flow of liquid oxygen through a turbine where real gas effects are significant. Vortex shedding predictions with the LOX formulations reduce the discrepancy between perfect gas computations and experiment by approximately an order of magnitude, thereby verifying the real gas formulation as well as providing an effective case where its capabilities are necessary.

Introduction

Turbomachinery devices are in use around the world in an almost uncountable number of applications. The working media employed in these diverse applications incorporate nearly every fluid known to man. There are turbomachinery implementations that deal with gases, liquids, cryogenics, slurries, multi-phase and multi-component fluids, supercritical fluids and more. There is perhaps no other class of engineering devices for which the diversity of fluids is so great.

Despite this diversity of working fluids, the primary emphasis of Computational Fluid Dynamics (CFD) applications in turbomachinery has been restricted to perfect gases, with a secondary emphasis on incompressible fluids. The primary reasons are that the field of CFD itself has been driven by aerospace applications in which perfect gases are the dominant fluid; and because the initial CFD emphasis in turbomachinery was on high technology, perfect gas applications such as aircraft gas turbines where the non-trivial development costs of code development could be more easily amortized.

The incompressible fluid codes that have been developed, have been largely independent of compressible flow codes and have used algorithms that are completely unrelated to compressible algorithms. As a consequence, the simulation of both a perfect gas and an incompressible fluid through a given turbomachine has typically required two completely different codes. This separation of codes and algorithms by specific fluid type is an issue of particular importance in turbomachinery where, as noted above, a broad spectrum of fluids is the rule, not the exception.

The turbomachinery in liquid propellant rocket engines constitutes a classic example of the global fluid diversity seen in the broader turbomachinery field. Liquid rocket engines typically pump or expand liquids, supercritical fluids, cryogenic fluids, hot combustion gases and multi-phase fluids, all in a single engine. This is a major contrast to aircraft engines where the working fluid is nearly always limited to air or vitiated air. In addition to a broad variety of fluids, the rocket engine also involves a more types of turbomachinery (centrifugal, axial, impellers, etc.) than do aircraft engines which are often limited to axial machines. This breadth of turbomachinery types dictates that the CFD codes for rocket turbomachinery be more general and more flexible, and again argues for a broad fluid capability in a single code, rather than requiring a different code for each fluid.

The purpose of the present paper is to outline a common algorithm for solving the equations of motion for a general fluid independent of its property characteristics. This technique has been incorporated in a new turbomachinery code that uses a common flow solver for the conservation relations and that relegates the fluid properties to subroutines specialized for the fluid of interest. Representative results for two- and three-dimensional applications are given to demonstrate capabilities for a perfect gas and for croygenic liquid oxygen with a highly complex equation of state.

The Conservation Equations for an Arbitrary Fluid

As noted above, what has classically become known as 'Computational Fluid Dynamics' has been largely limited to perfect gases with constant specific heats. If these highly developed algorithms are to be extended to a general fluid with arbitrary properties and an arbitrary equation of state, it is necessary first to identify those areas in which the current algorithms have been specialized to perfect gases and then take the steps necessary to extend them to arbitrary fluids.

The obvious first step is to write the conservation equations in a form that is applicable to a general fluid. This is easily accomplished by writing the equations in standard conservation form while retaining the enthalpy (as opposed to the specific heats) in the energy equation. The equations of motion for the steady or unsteady dynamics of a general, Newtonian fluid may be written in a generic space-time divergence form by first defining a four-dimensional, spatial-temporal divergence operator

$$\nabla = \frac{\partial}{\partial t} e_t + \frac{\partial}{\partial x} e_x + \frac{\partial}{\partial y} e_y + \frac{\partial}{\partial z} e_z \qquad (1)$$

where the quantities e_t, e_x, e_y and e_z, represent unit vectors in the temporal and spatial directions, respectively.

We next introduce a companion four-component flux tensor, F, for the inviscid flux,

$$F = Qe_t + Ee_x + Fe_y + Ge_z \qquad (2)$$

where the individual inviscid flux vectors are defined as,

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho h^0 - p \end{pmatrix} \quad E = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho uh^0 \end{pmatrix} \quad F = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho vh^0 \end{pmatrix} \quad G = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho wh^0 \end{pmatrix} \qquad (3)$$

Here, all quantities take on their standard notation.

We also define a degenerate, three-component tensor, F_v, for the viscous flux,

$$F_v = E_v e_x + F_v e_y + G_v e_z \qquad (4)$$

These viscous expressions reflect the absence of temporal derivatives in the diffusion terms and contain the standard stress tensor which, for brevity, is omitted here.

Combining Eqs. 1 through 4, allows us to express both the steady and unsteady forms of conservation equations for a general fluid by the general divergence expression:

$$\nabla \bullet (F - F_v) = 0 \qquad (5)$$

These conservation equations are not specialized to any particular fluid, but remain completely general. Multi-component and multi-phase fluids can also be described by this same system if additional conservation relations are added to describe the added physical complexities. An important consequence of this common pde format is that it should be possible to formulate numerical procedures in such a manner that the same pde solver can be used for any fluid, regardless of its characteristics.

Auxiliary Relations for the Fluid Properties

The set of conservation equations in Eq. 5 is not closed, but requires additional algebraic relations to specify the fluid properties. In particular, we must augment the divergence expression in Eq. 5 by complementary relations for the thermodynamic and caloric equations of state as well as appropriate relations for the transport properties. (We must also specify pertinent differential and/or algebraic equations for turbulence modeling, but these are omitted here.)

To retain an ability to solve general fluids, we express these auxiliary relations as arbitrary functions of pressure and temperature. The thermodynamic and caloric equations of state are,

$$\rho = \rho(p,T) \qquad (6)$$

and,

$$h = h(p,T) \qquad (7)$$

while the stagnation enthalpy, h^0, is defined as,

$$h^0 = h + (1/2)u_i^2 \qquad (8)$$

Transport properties, including viscosity, thermal conductivity and mass diffusivity can similarly be written as arbitrary functions of pressure and temperature. These property characteristics can be expressed as closed-form relations, table look-up format, or other means, but when combined with the divergence expression in Eq. 5 they completely define the equation set. We are now in a position to seek the numerical solution of this system.

Discretization, Artificial Dissipation and Convergence

The numerical solution of the equations of motion requires that Eq. 5 be discretized to convert it to an algebraic system. Available discretization procedures include finite difference, finite volume and finite element methods. Here we choose a finite volume method that defines the 'numerical' flux on each cell face by means of a Riemann solver [xx]. The implementation of the Riemann solver requires that a pseudo-time derivative be added to Eq. 5. The solution of the resulting discretized system then requires an iterative method. To implement the iterative procedure we add a second pseudo-time derivative and use a time-marching procedure. The divergence form of the equations (Eq. 5) suggests that a pseudo-time can be used for both steady and unsteady equations in the same manner.

Although the Riemann solver and the iterative process are generally accomplished with a single time derivative, we separate them and perform the two steps sequentially using a distinct time for each process. This separation helps to distinguish the critical issues. Note that the time derivative used for either of these processes does not represent physical time, but rather an artificial time. To distinguish this pseudo time from the physical time that appears inside the divergence operator, we designate the pseudo-time by τ.

The pseudo-time derivative that is added to Eq. 5 must be dimensionally consistent with the divergence operator. Dimensional consistency can be insured by using the conservative variables in the pseudo-time derivative, $\partial Q/\partial\tau$, but since this artificial time derivative goes to zero as the solution is approached, there is no advantage to using conservative variables. A non-conservative vector will work just as well. To retain dimensional consistency, we transform the pseudo-time derivative to the more convenient primitive variables, $Q_p = (p,u,v,w,T)^T$ by means of the chain rule:

$$\frac{\partial Q}{\partial \tau} = \frac{\partial Q}{\partial Q_p}\frac{\partial Q_p}{\partial \tau} \equiv \Gamma_p \frac{\partial Q_p}{\partial \tau} \qquad (9)$$

The equations of motion then take the 'time-marching' form:

$$\Gamma_p \frac{\partial Q_p}{\partial \tau} + \nabla \bullet (F - F_v) = 0 \qquad (10)$$

It is clear that in the limit as τ becomes infinite and the pseudo-time derivative approaches zero that Eq. 10 reduces to Eq. 5. We can therefore replace the artificial time derivative coefficient, Γ_p, by an alternative matrix that ensures accuracy or improved convergence, depending on whether we are working with the artificial dissipation step or the time-marching step. Note that primitive variables are consistent with our thermodynamic and caloric equations of state in Eqs. 6 and 7.

In the following two subsections we perform the numerical discretization and then define a time-marching procedure.

Upwind Discretization and Artificial Dissipation

Numerical schemes for hyperbolic systems require artificial dissipation. When an approximate Riemann solver is used to generate an upwind method, the artificial dissipation is an inherent part of the discretization, but this does not ensure that it will have an acceptable magnitude. The artificial dissipation added by an upwind method depends upon the flow conditions and the fluid properties as determined from Eqs. 6 and 7. These characteristics must be taken into account in both the discretization step and the convergence algorithm. To define the Riemann solver, we denote the pseudo time by the variable, τ_1 (where the subscript 1 implies that this is the first pseudo time), and the

associated coefficient matrix by Γ_1 implying that each pseudo time may have a distinct coefficient matrix.

Using this notation, we initiate the discretization process in standard fashion by integrating Eq. 10 over a control volume. The divergence term is immediately converted to a four-dimensional surface integral by the divergence theorem, but the volume integral of the pseudo-time term cannot be integrated directly because it is in non-conservative form. To circumvent this, must convert the time derivative to a perfect differential form. As is seen below, it is necessary to replace Γ_1 by an artificial matrix for some equations of state and some flow conditions so that $\Gamma_1 \neq \partial Q / \partial Q_p$. We therefore define a new differential quantity, $\partial Q' = \Gamma_1 \partial Q_p$, and compute the volume integral of the temporal derivative as,

$$\oint_{\Omega} \Gamma_1 \frac{\partial Q_p}{\partial \tau_1} d\Omega = \oint_{\Omega} \frac{\partial Q'}{\partial \tau_1} d\Omega = \Omega \frac{\partial \overline{Q'}}{\partial \tau_1} \qquad (11)$$

where Ω is the volume and $\overline{Q'}$ is the average value of Q' over the volume.

Upon replacing the surface integral by a summation over K faces, the finite volume expression becomes,

$$\frac{\partial \overline{Q'}}{\partial \tau_1} \Omega + \sum_{k=1}^{K} \hat{F}_k \bullet \mathbf{n}_k = 0 \quad (12)$$

where $\hat{F}_k$ represents the numerical flux vector on the k^{th} face, and the normal vector, $\mathbf{n}_k$, includes both the magnitude and direction of the normal to the face. Note that $\hat{F}_k$ includes both spatial and temporal (pseudo-time) components. We need to switch back to primitive variables, but first employ an approximate Riemann solver to define the fluxes on the surface.

Using standard procedures, the numerical flux, $\hat{F}_{nk}$, normal to the face k in Eq. 12 can be written as,

$$\hat{F}_{nk} \equiv \frac{1}{2}\left(F_{nL} + F_{nR}\right)_k - \frac{1}{2}\left|\frac{\partial F_n}{\partial Q_1'}\right|_k \left(\delta Q_1'\right)_k \quad (13)$$

where the subscripts R and L represent values on the right and left side of the face. Note that we must retain the variable, Q', in computing this flux, since it is the variable that appears in the time derivative of Eq. 12. Similarly, Q', is the variable that determines the artificial dissipation in the upwind formula.

Converting back to primitive variables using, $\Gamma_1 = \partial Q_1' / \partial Q_p$ and defining the Jacobian, $(A_p)_n = \partial F_n / \partial Q_p$ gives,

$$\hat{F} \equiv \frac{(F_L + F_R)}{2} - \Gamma_1 \left| \Gamma_1^{-1} A_p \right| \frac{(Q_{pR} - Q_{pL})}{2} \quad (14)$$

where we have dropped the subscripts, k and n to simplify the notation. The second term in Eq. 15 is the artificial dissipation term that was introduced by the approximate Riemann procedure. Note that the artificial dissipation, the upwind direction and the characteristics speeds are controlled by the pseudo time, τ_1, and the corresponding time-derivative matrix, Γ_1. From Eq. 14 it is clear that the inverse of the matrix, Γ_1, must exist. Additional considerations indicate that Γ_1 must be well conditioned to provide acceptable levels of artificial dissipation. Consequently, for the general case, Γ_1 must be an artificial matrix, not the physical Jacobian. Methods for determining the matrix are given in Ref. xx.

The above procedure gives a numerical representation for the inviscid fluxes at the cell faces. To these inviscid terms we must also add the viscous counterparts, but these require no special procedures and their development is omitted.

We now discretize Eq. 12 in pseudo-time, τ_1 and set the time step, $\Delta\tau_1$ to infinity so that the effects of this first pseudo-time vanish (apart from its role in defining the artificial dissipation). This gives an algebraic set of equations for the cell fluxes:

$$\frac{1}{\Omega} \sum_{k=1}^{K} \left(\hat{F}_k - \hat{F}_{vk} \right) \bullet \mathbf{n}_k = 0 \quad (15)$$

There is a flux balance equation of this form for each cell in the domain. Equation 15 represents a discretized version of the continuous divergence operator in Eq. 5 where each term in the discretized divergence operator is given by Eq. 14. Setting the time step, $\Delta\tau_1$, to zero enables us to separate the convergence process from the artificial dissipation.

Numerical Solution Algorithm

To solve the system in Eq. 15, we employ a second pseudo-time, τ_2 and a second time derivative matrix, Γ_2. This gives us the option of using a different matrix for convergence than for artificial dissipation, a step that can often be advantageous [xx]. We therefore wish to obtain the 'steady' solution to the equation system,

$$\Gamma_2 \frac{\partial Q_p}{\partial \tau_2} \Omega + \sum_{k=1}^{K} \left(\hat{F} - \hat{F}_v \right)_k \bullet \mathbf{n}_k = 0 \quad (16)$$

Note that even for transient problems we seek the steady solution to Eq. 16 because the physical time derivative appears inside the numerical divergence operator.

For steady solutions, the Euler implicit method is preferred, and upon combining Euler implicit discretization in pseudo time, τ_2, with the above discretization in space-time, we obtain:

$$\bar{\Gamma}_2 \frac{Q_p^{m+1} - Q_p^m}{\Delta\tau_1} \Omega + \sum_{k=1}^{K} \left(\hat{F} - \hat{F}_v\right)_k^{m+1} \bullet \mathbf{n}_k = 0 \quad (17)$$

We solve this equation by linearizing, placing it in delta form and solving by an appropriate approximate factorization method such as ADI, LGS, LU or etc. [xx,yy]. For the examples given here, we have used a line Gauss-Seidel method [xx]. The steps involved in solving Eq. 17 are standard, and are independent of the fluid characteristics, the flow conditions and the equation of motion. We choose the matrix, Γ_2, in such a fashion that it will lead to efficient convergence [xx]. Again, the discretized equations given in Eq. 17 along with the definition of the numerical fluxes in Eq. 14 and the properties relations in Eqs. 6 and 7 hold for any type of fluid. Consequently, it is reasonable to expect that a common and universal algorithm could be used to solve them.

Note that the flux vectors given in the equation system given in Eq. 17 include the physical time, t, while the pseudo-time, τ_2, occurs as a primary time-marching variable. Accordingly we can describe Eq. 17 as a 'dual'-time iterative method with the inner time beign represented by the psuedo-time, τ_2, and the outer time by the physical time, t.

Boundary Condition Specification

The application of boundary conditions is crucial for any CFD problem. Here we briefly summarize a technique for applying upstream subsonic boundary conditions for an arbitrary fluid. Supersonic inflow and subsonic outflow procedures are analogous to perfect gas methods and can be deduced from the present discussion. As a boundary condition specification, we assume a cell centered finite volume method that makes use of ghost cells.

For a real gas, the familiar upstream boundary conditions of constant stagnation pressure and constant stagnation enthalpy are not physically correct and should be replaced by the more fundamental constant entropy and constant stagnation enthalpy conditions. The application of these real gas boundary conditions is analogous to the method used for perfect gases, but the implementation is different. As companion boundary conditions, we specify two flow angles, and extrapolate the magnitude of the velocity, q, from conditions inside the computational domain.

The four subsonic upstream boundary conditions are

$$s = s_{ref}, \quad h^0 = h^0_{ref} \qquad (18)$$
$$v/u = (v/u)_{ref}, \quad w/u = (w/u)_{ref},$$

The principal issue is to find appropriate relations to determine the change in flow variables (here, the primitive variables) in the ghost cell. To do so, we define a boundary condition vector, Ω that contains the variables in the boundary condition plus a complementary vector that contains the extrapolated information,

$$\Omega_{bc} = \begin{pmatrix} s_{ref} \\ 0 \\ (v/u)_{ref} \\ (w/u)_{ref} \\ h^0_{ref} \end{pmatrix} \quad \Omega_{eq} = \begin{pmatrix} 0 \\ q_{char} \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad (19)$$

The subscript, *bc*, implies that these four conditions are boundary conditions, whereas the subscript, *eq*, indicates that this one condition is obtained from the equations of motion. It may be extrapolated (as indicated above) or may be a characteristic condition.

We then define a ghost cell vector,

$$\Omega_g = \begin{pmatrix} s \\ q \\ v/u \\ w/u \\ h^0 \end{pmatrix} \qquad (20)$$

and enforce conditions in the ghost cell by considering these three vectors as functions of the primitive variables, Q_p. We first equate the sum of the vectors in Eq. 19 with the vector in Eq. 20, $\Omega_{re} = \Omega_{bc} + \Omega_{eq}$, and use a Taylor's series expansion to enforce this implicitly at the new time level. The Taylor's series gives,

$$\left(\frac{\partial \Omega_g}{\partial Q_{pg}}\right)\Delta Q_{pg} - \sum_{i=1}^{N}\left\{a_i\left(\frac{\partial \Omega_{eq}}{\partial Q_{pi}}\right)\Delta Q_{pi}\right\} = -\Omega^n_g + \Omega^n_{bc} + \sum_{i=1}^{N} a_i \Omega^n_{eq} \qquad (21)$$

where the summation goes over those grid points used to extrapolate the velocity magnitude or to define the characteristic relation and the a_i's represent influence coefficients for the representative coefficients. Equation 21 constitutes an update equation for the change in the primitive variables in the ghost cell. The Jacobian matrices are given in the Appendix.

Computational Results

The Generalized Equation Set given above has been implemented into existing two-dimensional (Wildcat) and three-dimensional (Corsair) turbomachinery codes [1]. The new two-dimensional code is called Aardvark, while the three-dimensional code is called Phantom. The real fluid models implemented in the codes were provided by Oefelein [2]. The capabilities of the GES are demonstrated using three test cases. The first test case involves the flow of air in a 1-1/2 stage, low-speed, large-scale turbine rig for which there exists a substantial experimental data base. The second test case is for the flow of air in a single-stage supersonic turbine. The last test case is of liquid oxygen (LOX) flowing in a hydraulic turbine, which was recently the subject of an extensive analysis focusing on vortex shedding from the vane trailing edge.

All three simulations were run on an SGI Origin2000 with 32 195-MHz processors.

Large-Scale Rotating Rig

The Large-Scale Rotating Rig (LSRR) turbine facility at United Technologies Research Center was used for many years as a test bed for studying turbine flow physics [3]. The 1-1/2 stage configuration of the LSRR with 50% (of the first vane chord) axial gaps is employed for the current study. The inlet Mach number to the turbine is M=0.07, the inlet pressure and temperature are 14.7 psia and 530 deg R, respectively, and the ratio of turbine exit static pressure to turbine inlet total pressure is 0.9505. The turbine rotates at 410 RPM. The actual turbine rig contains 22 first-stage vanes, 28 first-stage rotors and 28 second-stage vanes. For the purposes of this investigation equal blade counts were assumed in all three rows, allowing a 1-vane/1-rotor/1-vane simulation, and the first-stage vanes were scaled by a factor of 22/28 to maintain pitch-to-chord ratio, blockage and mass flow. The objective of the first test case is to validate the codes against a baseline turbine data set.

The two-dimensional computational grid for the turbine rig, which contains 46,000 grid points, is shown in Fig. 1. The average value of y+, the non-dimensional distance of the first grid point off the airfoil surface, was approximately 1.0 for all three rows. In addition, the surface boundary layers were discretized with 15-20 grid points. The computational grid for the three-dimensional simulation (which was run for only the first stage of the turbine) contains 21 spanwise planes and 372,000 grid points. Both the two- and three-dimensional test cases were run using dual time stepping with two inner iterations and steady preconditioning.

Figure 2 contains instantaneous non-dimensional static pressure contours in the turbine obtained with Aardvark (2D code), while Fig. 3 contains the midspan contours obtained with Phantom (3D code). In both figures the local pressure is non-dimensionalized by the inlet static pressure. The flow reaches a Mach number of 0.23 as it accelerates through the turbine, and the pressure contours give an indication of vortex shedding from

the trailing edges of the airfoils. This figure also highlights the relatively smooth transfer of data between adjacent h- and o-type grids. Instantaneous non-dimensional entropy contours predicted using Aardvark are presented in Fig. 4. Each of the three blade rows exhibit strong trailing-edge vortex shedding, which is common in low-speed turbines. Figure 5 shows a comparison of the predicted and experimental time-averaged airfoil loadings. The pressure coefficient in Fig. 5 is calculated as:

$$C_p = (P-P_{exit})/(Pt_{in}-P_{ex}) \qquad (22)$$

In general, the predicted results display good agreement with the experimental data. The differences on the suction surface of the rotor are characteristic of other comparisons found in the literature.

Single-Stage Supersonic Turbine

The single-stage supersonic turbine configuration is typical of those proposed for a reusable launch vehicle. The turbine design has 21 vanes and 52 rotors. In the current effort a 20-vane/50-rotor airfoil approximation has been made, resulting in a simulation blade count of 2 vanes and 5 rotors. To keep the pitch-to-chord ratio (blockage) constant, the first-stage vanes were scaled by a factor of 21/20 and the rotor blades were scaled by a factor of 52/50. The objective of the second test case is to compare the results obtained from the new code using perfect and real gas assumptions with results from the perfect-gas, density-based code of Ref. [1]. This test case was run using the Aardvark (2-D) code only.

The computational grid for the turbine contained 60,000 grid points and is shown in Fig. 6. As in the first test case, the average value of y^+, the non-dimensional distance of the first grid point off the airfoil surface, was approximately 1.0 for all three rows. In addition, the surface boundary layers were discretized with 10 to 15 grid points. The test case was run using dual time stepping with two inner iterations and no preconditioning.

The turbine has a design inlet Mach number of $M = 0.25$, an inlet static pressure of 2225 psia, and an inlet static temperature of approximately $T = 2225$ R. The rotor rotates at 32,000 RPM, the Reynolds number based on the inlet conditions and the rotor axial chord is approximately 1.2×10^6 and the ratio of the rotor exit static pressure to vane inlet total pressure is 0.125. The operating fluid is gaseous oxygen, here treated as a real gas although the in the operational regime, it behaves nearly like a perfect gas.

Figure 7 illustrates instantaneous Mach contours in the turbine. The flow enters the vane passage at a low subsonic Mach number. The flow accelerates through the vane passage, reaching sonic (choked) conditions at the throat. The flow becomes supersonic in the diverging portion of the vane passage, reaching a peak of nearly Mach 2.0. A strong expansion wave system emanates from the pressure side of the trailing edge. The expansion waves move across the vane passage and interact with the suction surface of the adjacent vane. The vane wakes convect into the rotor passage where they interact

with the rotor bow shock. This interaction creates a subsonic flow region near the leading edge that weakens the boundary layer and causes rotor suction surface boundary layer separation. This condition exists until the rotor passes through the vane wake. The flow diffuses as it exits the rotor passage, decelerating back to subsonic flow conditions.

Comparisons of the predicted time-averaged vane and rotor pressure loadings are shown in Figs. 8 and 9, respectively. The pressure on the pressure surface of the vane remains relatively constant until the uncovered portion of the passage is reached, at which point the flow rapidly accelerates. The suction side flow is characterized by four different phenomena: 1) acceleration of the flow up to the throat (located at approximately x/c = 0.44), 2) a small region of diffusion, 3) additional acceleration of the flow in the region where the expansion fan from the adjacent blade interacts with the surface (located at approximately x/c = 0.65), and 4) a small area of diffusion, then constant pressure until the trailing edge is reached. In general the perfect gas results from the new code (Aardvark) show excellent agreement with the results of the density-based code (Wildcat). The only discrepancies occur on the suction surface of the vane, where Aardvark predicts the interaction of the expansion wave with the surface to be somewhat downstream of where it is predicted by Wildcat. The use of the real gas model did little to modify the results of the perfect gas simulation. The pressure on the pressure surface of the rotor is nearly constant along its length. The suction surface loading shows the interaction with bow shock of the adjacent rotor (approximately x/c = 0.20), followed by flow acceleration that is terminated by a weak shock (x/c = 0.80). There is generally excellent agreement between the results predicted by the two codes, except that Aardvark predicts the interaction with the bow shock to be further downstream. Again, the real gas effects were minimal.

Low-Pressure Oxidizer Turbine

The last test case involves a more significant application of the GES methodology. Recently, cracks were discovered on a low-pressure oxidizer turbine in service. An investigation was initiated to determine the source of the cracks [4]. Although the operating fluid of the turbine is liquid oxygen (LOX), the Wildcat (perfect gas) code was initially used to perform simulations. The results of the Wildcat simulations suggested that vortex shedding might be the main aerodynamic driver in the problem. The vortex shedding frequency was predicted to be 43.8 *kHz*, while a re-assessment of previous engine data suggested the shedding frequency to be around 36.0 *kHz*. In an effort to improve the fidelity of the numerical predictions the Aardvark and Phantom codes have been applied to the problem.

The oxidizer turbine design has 39 vanes and 61 rotors. In the current effort a 40-vane/60-rotor airfoil approximation has been made, resulting in a simulation blade count of 2 vanes and 3 rotors. To keep the pitch-to-chord ratio (blockage) constant, the first-stage vanes were scaled by a factor of 39/40 and the rotor blades were scaled by a factor of 61/60. The computational grid for the turbine contained 57,000 grid points and is shown in Fig. 10. The average value of y^+, the non-dimensional distance of the first grid point off the airfoil surface, was approximately 1.5 for all three rows. In addition, the

surface boundary layers were discretized with 10-15 grid points. The computational grid for the three-dimensional simulation contains 21 spanwise planes and 1.14 million grid points. Both test cases were run using dual time stepping with two inner iterations and steady preconditioning.

The turbine has a design inlet Mach number of $M = 0.04$, an inlet static pressure of 4000 psia, and an inlet static temperature of approximately $T = 189$ R. The rotor rotates at 5169 RPM, the Reynolds number based on the inlet conditions and the rotor axial chord is approximately 0.5×10^6 and the ratio of the rotor exit static pressure to vane inlet total pressure is 0.788. The operating fluid is LOX.

Instantaneous non-dimensional pressure contours predicted using Phantom are shown in Fig. 11. The pressure field is characterized by strong acceleration in the vane passage, followed by more moderate acceleration in the rotor passage. The flow reaches a peak Mach number of approximately 0.20. Instantaneous non-dimensional velocity contours predicted using Aardvark are shown in Fig. 12. The velocity contours indicate passage-to-passage variations in the flow field, as well as vortex shedding from the vane. The vortex shedding is highlighted in Fig. 13, which shows non-dimensional temperature contours. The shed vortex street is convected into the rotor passage, where it interacts with the passing rotor blades. The predicted shedding frequency using Aardvark is 36.9 kHz, 36.5 kHz using Phantom and, as noted above, 43.8 kHz using the Wildcat code. Thus, using the real fluid properties results in a predicted vortex shedding frequency that is much closer to the experimentally deduced value.

References

1. Dorney, D. J., Davis, R. L., Edwards, D. E. and Madavan, N. K., ``Unsteady Analysis of Hot Streak Migration in a Turbine Stage,'' AIAA Journal of Propulsion and Power, Vol. 8, No. 2, March-April, 1992, pp. 520-529.

2. Oefelein, J. C., Private Communication, December, 2002.

3. Dring, R. P., Blair, M. F., Joslyn, H. D., Power, G. D., and Verdon, J. M., "The Effects of Inlet Turbulence and Rotor/Stator Interactions on the Aerodynamics and Heat Transfer of a Large-Scale Rotating Turbine Model. I – Final Report," NASA CR 4079, Contract NAS3-23717, May, 1986.

4. Nesman, T. E., "Rocket Engine Oscillation Diagnostics," The 2002 International Congress and Exposition on Noise Control Engineering, Paper Number N-235, Dearborn, MI, USA. August 19-21, 2002.

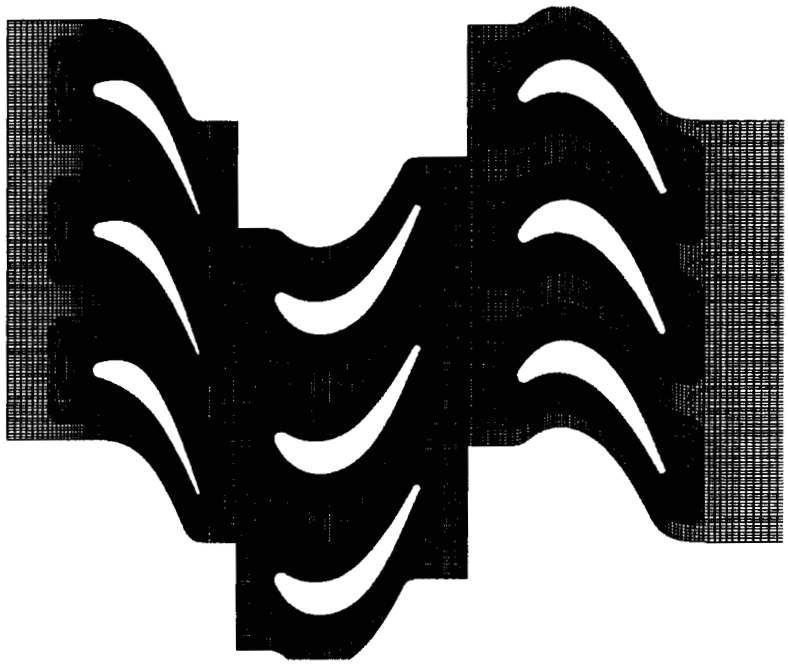


Figure 1. Computational grid for the LSRR turbine rig.

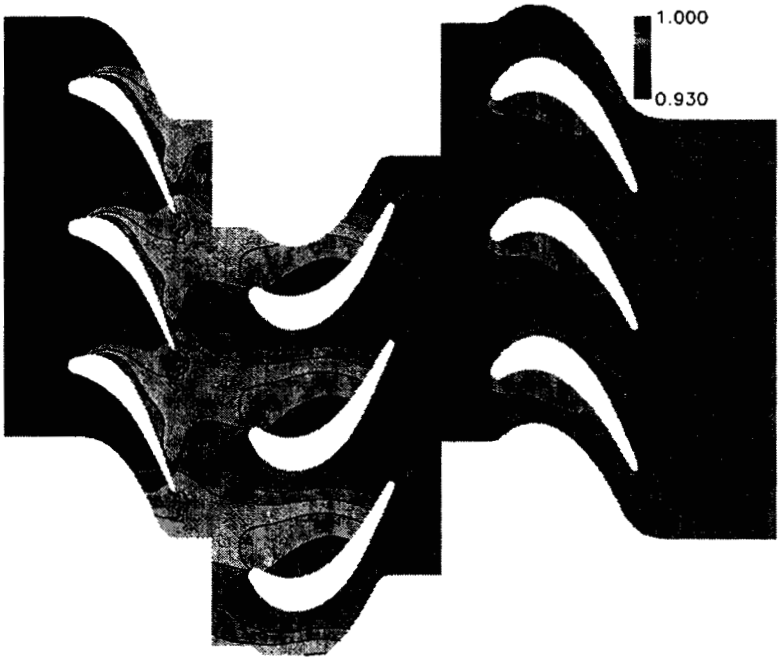


Figure 2. Instantaneous non-dimensional pressure contours for the LSRR turbine - Aardvark.

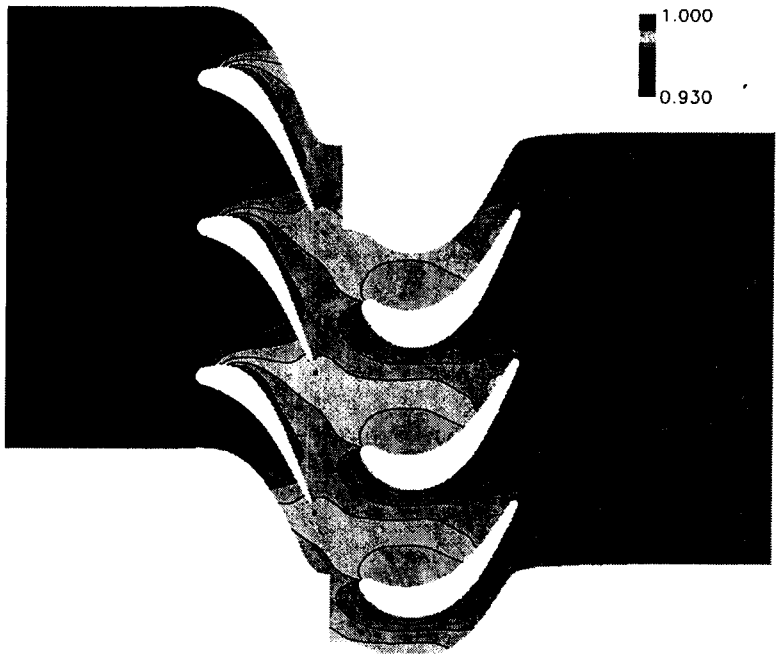


Figure 3. Instantaneous non-dimensional pressure contours for the LSRR turbine – midspan -Phantom

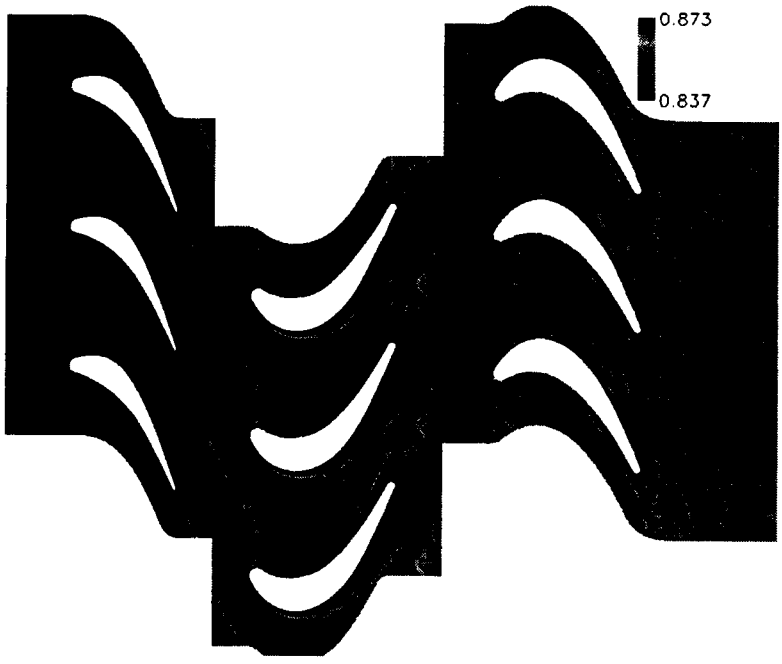
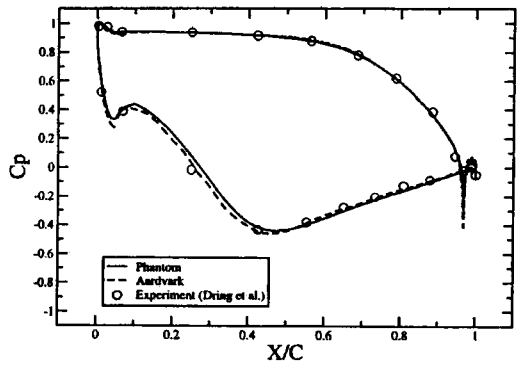
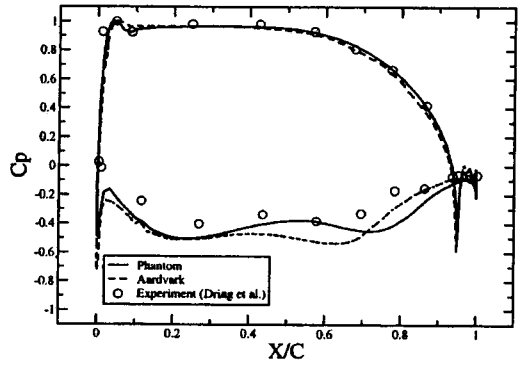


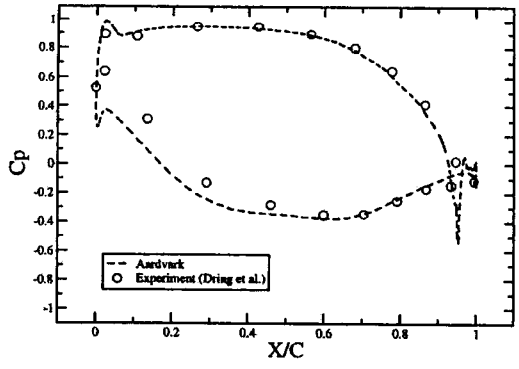
Figure 4. Instantaneous non-dimension entropy contours for the LSRR turbine - Aardvark.



Vane -1



Rotor



Vane-2

Figure 5. Time-averaged blade loadings for the LSRR turbine.

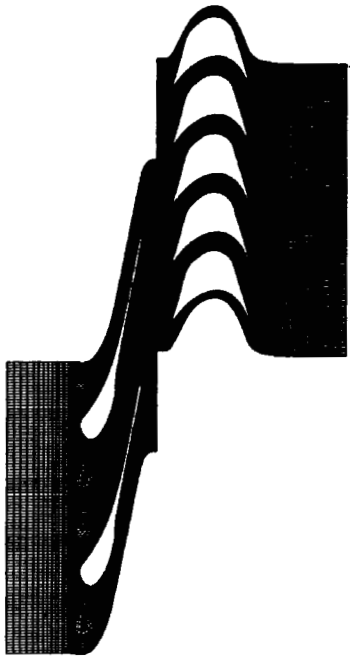


Figure 6. Computational grid for the supersonic turbine.

Figure 7. Instantaneous Mach contours in the supersonic turbine - Aardvark.

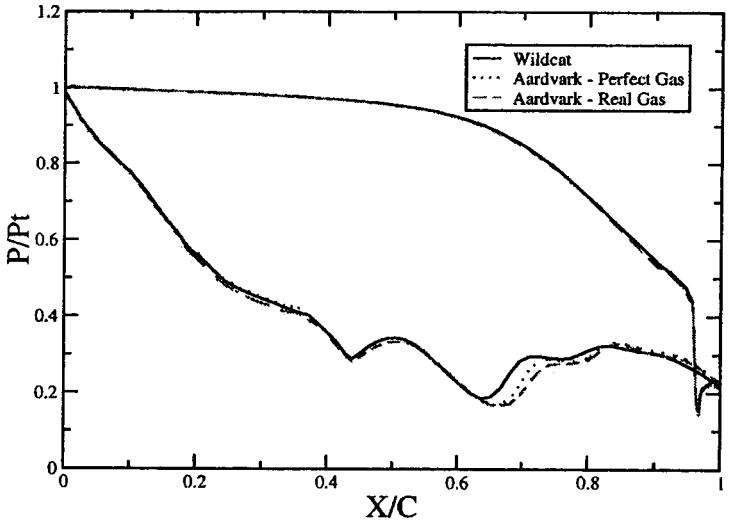


Figure 8. Time-averaged blade loadings on the vane of the supersonic turbine.

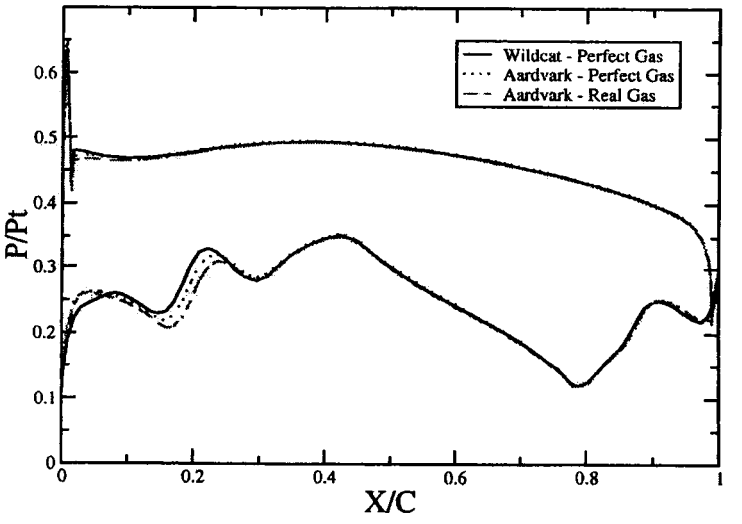


Figure 9. Time-averaged blade loadings on the rotor of the supersonic turbine.

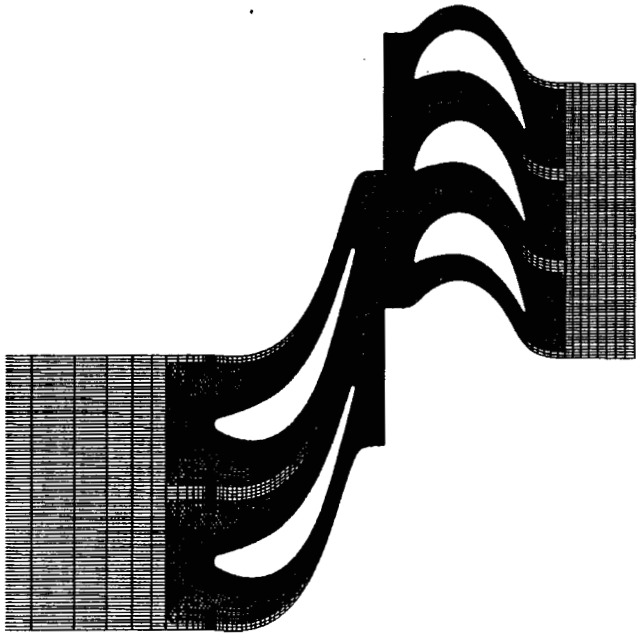


Figure 10. Computational grid for the oxidizer turbine - Aardvark.

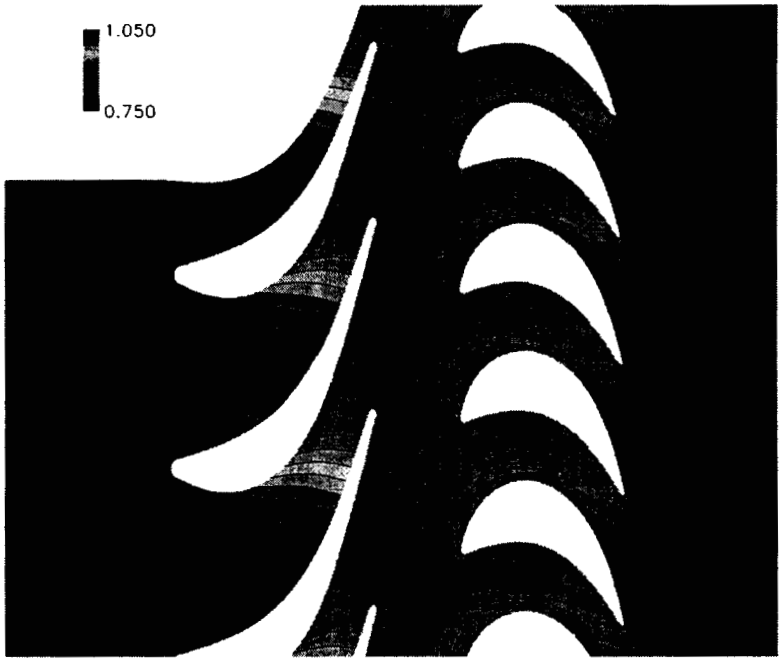


Figure 11. Instantaneous non-dimensional pressure contours in the oxidizer turbine – midspan - Phantom.

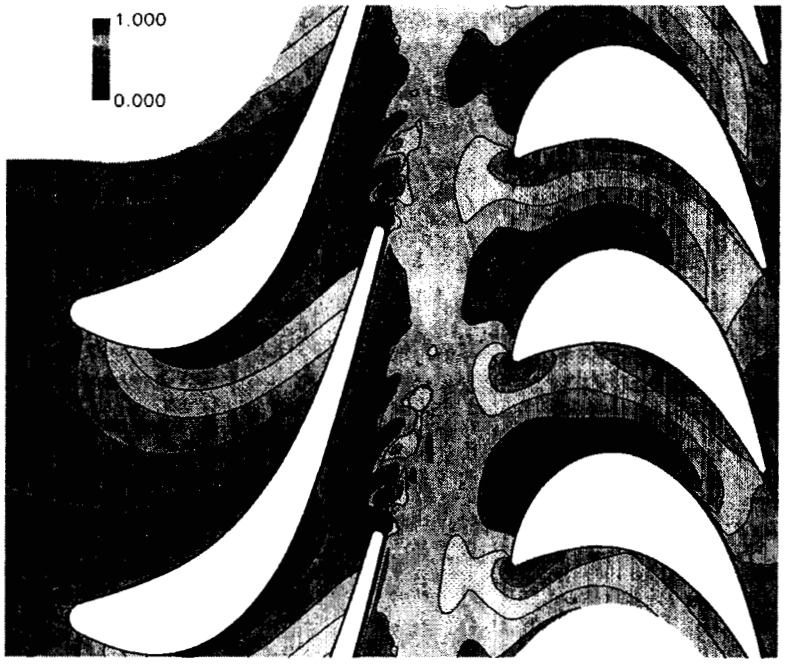


Figure 12. Instantaneous non-dimensional velocity contours in the oxidizer turbine - Aardvark.

Figure 13. Instantaneous non-dimensional temperature contours in the oxidizer turbine - Aardvark.

Appendix
Boundary Condition Jacobians for Real Gas

Pertinent relations for the boundary condition Jacobian in Eq. 21 can be deduced by starting from the general enthalpy relation in Eq. 7 and the differential expression for entropy in terms of pressure and temperature,

$$Tds = dh - \frac{1}{\rho}dp = -(1-\rho h_p)\frac{dp}{\rho} + h_T dT \quad \text{(A.1)}$$

By comparing this with the differential for entropy expressed as a function of pressure and temperature,

$$ds = \left(\frac{\partial s}{\partial T}\right)_p dT + \left(\frac{\partial s}{\partial p}\right)_T dp \quad \text{(A.2)}$$

we can write the partial derivatives of entropy as,

$$\left(\frac{\partial s}{\partial p}\right)_T = \frac{-(1-\rho h_p)}{\rho T}$$
$$\left(\frac{\partial s}{\partial T}\right)_p = \frac{h_T}{T} \quad \text{(A.3)}$$

Similarly, the differential of the stagnation enthalpy is,

$$dh^0 = h_p dp + h_T dT + u du \quad \text{(A.4)}$$

The derivatives of the flow angles are:

$$\frac{\partial(v/u)}{\partial u} = -\frac{v}{u^2} \text{ and } \frac{\partial(w/u)}{\partial u} = \frac{1}{u} \quad \text{(A.5)}$$

with similar relations for the other velocity components. The combination of Eqs. A.2 through A.4 aalong with Eq. A.5 allows us to determine the Jacobian matrix,

$$\frac{\partial \Omega_g}{\partial Q_p} = \begin{pmatrix} -\frac{1-\rho h_p}{\rho T} & 0 & 0 & 0 & \frac{h_T}{T} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{v}{u^2} & \frac{1}{u} & 0 & 0 \\ 0 & -\frac{w}{u^2} & 0 & \frac{1}{u} & 0 \\ h_p & u & v & w & h_T \end{pmatrix} \quad \text{(A.6)}$$

Note that we have placed the null equation in the second row, and that this row remains zero.

The Jacobian inside the summation in Eq. 21 contains only one row. For the extrapolation of the velocity magnitude, we have for a multi-point extrapolation,

$$q_{extrap} = \sum_{i=1}^{N} a_i q_i \qquad \text{(A.7)}$$

where a_i represents the weighting coefficients for the extrapolation, and q_i represents the velocity magnitude in the N cells that are used for the extrapolation.

The derivatives of q and the flow angles are:

$$\frac{\partial q}{\partial u} = \frac{\partial \sqrt{u^2 + v^2 + w^2}}{\partial u} = \frac{u}{\sqrt{u^2 + v^2 + w^2}} \qquad \text{(A.8)}$$

with similar relations for v and w. The derivatives of the flow angles are:

$$\frac{\partial (v/u)}{\partial u} = -\frac{v}{u^2} \quad \text{and} \quad \frac{\partial (w/u)}{\partial u} = \frac{1}{u} \qquad \text{(A.9)}$$

Placing these inside the Jacobian matrix, we have:

$$\left(\frac{\partial q \Omega_{\theta\_ref}}{\partial Q_p} \right) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{u}{q} & \frac{v}{q} & \frac{w}{q} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \text{(A.10)}$$