Formulation and Implementation of Inflow/Outflow Boundary Conditions to Simulate Propulsive Effects

David L. Rodriguez[*]
Science & Technology Corp., Moffett Field, CA 94035

Michael J. Aftosmis[†] and Marian Nemec[‡]
NASA Ames Research Center, Moffett Field, CA 94035

Boundary conditions appropriate for simulating flow entering or exiting the computational domain to mimic propulsion effects have been implemented in an adaptive Cartesian simulation package. A robust iterative algorithm to control mass flow rate through an outflow boundary surface is presented, along with a formulation to explicitly specify mass flow rate through an inflow boundary surface. The boundary conditions have been applied within a mesh adaptation framework based on the method of adjoint-weighted residuals. This allows for proper adaptive mesh refinement when modeling propulsion systems. The new boundary conditions are demonstrated on several notional propulsion systems operating in flow regimes ranging from low subsonic to hypersonic. The examples show that the prescribed boundary state is more properly imposed as the mesh is refined. The mass-flow-rate steering algorithm is shown to be an efficient approach in each example. To demonstrate the boundary conditions on a realistic complex aircraft geometry, two of the new boundary conditions are also applied to a modern low-boom supersonic demonstrator design with multiple flow inlets and outlets.

Nomenclature

A	=	area of a boundary surface
c	=	speed of sound
$\underline{E}$	=	discrete flow total energy
$\overrightarrow{F}$	=	flux density tensor
H_t	=	total enthalpy
J^+, J^-	=	Riemann invariants
M_∞	=	Mach number of the freestream flow
$\dot{m}$	=	mass flow rate
$\hat{n}$	=	unit surface normal vector
u, v, w	=	discrete flow velocities in the Cartesian directions
p	=	discrete flow pressure
p_t	=	total pressure
$\underline{Q}$	=	discrete flow state vector of conservative variables $[\rho, \rho u, \rho v, \rho w, \rho E]^T$
$\tilde{\underline{Q}}$	=	integrated average of flow state in a control volume
R	=	constant in the perfect gas law
S	=	entropy
$\mathcal{S}_{\partial\Omega}$	=	surface area of the boundary of a control volume
t	=	time
T_t	=	total temperature
V	=	velocity magnitude
V_n	=	velocity component normal to a surface
V_t	=	velocity component tangent to a surface

[*] Senior Research Scientist, Computational Aerosciences Branch, MS 258-5; david.l.rodriguez@nasa.gov, Senior Member AIAA.
[†] Aerospace Engineer, Computational Aerosciences Branch, MS 258-5; michael.aftosmis@nasa.gov, Associate Fellow AIAA.
[‡] Aerospace Engineer, Computational Aerosciences Branch, MS 258-5; marian.nemec@nasa.gov, Senior Member AIAA.

$\mathcal{V}_\Omega$	=	volume of a computational domain
γ	=	ratio of specific heats
ρ	=	discrete flow density
$\partial\Omega$	=	boundary of a control volume or computational domain
$(\ )_b$	=	subscript indicating flow quantity at the computational domain boundary
$(\ )_i$	=	subscript indicating flow quantity from the computational domain interior
$(\ )_{set}$	=	subscript indicating flow quantity set by the user

I. Introduction

The proper simulation of propulsion effects is often essential to obtaining valuable results from high-fidelity, computational aerodynamic analyses. Properly modeling the stream tube entering an air-breathing engine or the plume exiting a nozzle can be crucial for accurate simulations. The Cartesian-mesh aerodynamics simulation package, Cart3D,[1] was originally developed with only a solid surface boundary condition along with a handful of farfield boundary treatments. Pandya et al.[2] later implemented a straightforward way of modeling domain inflow and outflow through portions of the model surface. This initial implementation requires the user to provide an entire flow state (ρ, u, v, w, p) at the boundary, and a Riemann solver is used to determine the actual boundary flux density tensor. Pandya verified and validated the implementation on a broad set of problems. While this boundary condition uses a very simple formulation that can model an extensive number of boundary states, it does often demand that the user know, a priori, the state of the flow leaving the domain at outflow boundaries or the state of the flow ahead of the domain at inflow boundaries. Knowledge of the boundary flow state is often not easily determined before a simulation and consequently must be iteratively determined increasing user burden. Moreover, this formulation lacks direct control over mass flow rate which is a critical parameter when simulating propulsive effects.

The work presented in this paper introduces new boundary condition implementations that extend the current capabilities of Cart3D. Four boundary conditions are presented: two for subsonic inflow and two for subsonic outflow. Two of these boundary conditions also allow mass-flow-rate control through a boundary surface. These new formulations relax the requirement that the complete boundary state be specified by the user. These boundary conditions have been linearized and implemented in the accompanying adjoint flow solver.[3] This allows the estimation of discretization error in user-selected output, such as lift or drag, when these boundary conditions are active, and enables adaptive mesh refinement leading to greater accuracy in outputs of interest.

II. Methodology

The three-dimensional Euler equations governing inviscid flow of a perfect gas may be written for a control volume Ω with closed boundary $\partial\Omega$ as:

$$\iiint_{\Omega} \frac{\partial}{\partial t} Q \, d\mathcal{V}_\Omega = - \oint_{\partial\Omega} (\vec{F} \cdot \hat{n}) \, d\mathcal{S}_{\partial\Omega} \tag{1}$$

where Q is the state vector of conserved variables and $\hat{n}$ is an outward facing unit vector. $\vec{F}$ is the tensor of flux density with components in all three Cartesian directions. The mean-preserving, cell-centered implementation locates the state vector at the centroid of all control volumes (cells) in the computational mesh. Assuming control volumes with planar faces that are fixed in time, and applying a midpoint quadrature rule, gives the semi-discrete form of the governing equations,

$$\frac{d}{dt}\tilde{Q} = -\frac{1}{\mathcal{V}_\Omega} \sum_{faces} \vec{F} \cdot \hat{n} \mathcal{S}_j \tag{2}$$

where j is a running variable over the faces of a control volume and $\tilde{Q}$ is now the integral cell average of the state.

The Cart3D analysis package uses a Cartesian cut-cell approach[4] in which the Euler equations are discretized on a multilevel Cartesian mesh with embedded boundaries. The mesh consists of regular Cartesian hexahedra everywhere, except for a layer of body-intersecting boundary cells. The spatial discretization uses a second-order accurate finite volume method with a weak imposition of boundary conditions. The flux-vector splitting approach of van Leer[5] is employed. Steady-state flow solutions are obtained using a five-stage Runge–Kutta scheme with local time-stepping and multigrid. Domain decomposition via space-filling curves permits parallel computation.[6]

Although the mesh consists of nested Cartesian cells, it is viewed as an unstructured collection of control volumes making the approach well-suited for solution-adaptive mesh refinement. The mesh refinement is based on a duality-preserving discrete adjoint solver.[7] While originally developed for gradient-based shape optimization,[3] the method is also exploited for error estimation and adaptive mesh refinement.[8] The adjoint-based error-estimation tailors the mesh refinement to reduce discretization error in a user-selected output of interest such as lift or drag. Error in a functional can be either driven below some pre-specified value, or alternatively, reduced as much as possible using a “worst-errors-first strategy” until reaching a limit in the total number of cells. Typically, adaptive simulations cost 2-4 times that of a single flow solve on the final mesh.

Two of the newly implemented boundary conditions presented in this work apply exclusively to subsonic outflow from the domain. The characteristic equations for this type of three-dimensional flow require exactly one flow attribute be set at the boundary interface. The other four attributes are determined by the state of the flow exiting the domain. For the first boundary condition, the pressure across the boundary is set to a constant value. For the second boundary condition, the velocity normal to the surface is fixed to a constant value. As will be shown, this second condition is very amenable to controlling the mass flow rate through the boundary. Note that whether outflow pressure or velocity is specified at the boundary, both approaches can be shown to be well-posed and result in viable simulation techniques.[9–11]

The other two boundary conditions presented are strictly for subsonic inflow into the computational domain. For these boundary conditions, only one flow attribute is propagated from the flow to the boundary and the other four attributes are set at the boundary itself. The first of these boundary treatments allows the user to set the total pressure and total temperature of the inflow. The flow is also implicitly set to enter the domain normal to the boundary surface, which effectively sets the other two flow attributes in three-dimensional flow. The second boundary condition allows the user to specify the total temperature and the mass flow rate, while the direction of the flow is again fixed to be normal to the boundary surface. These inflow and the outflow boundary conditions are presented in more rigorous detail in Sections A-D. Mass-flow-rate “steering” is discussed in Section E.

A. Back Pressure Subsonic Outflow

One of the more common subsonic outflow boundary conditions that is employed in numerous computational fluid dynamics solvers imposes a constant back pressure. This boundary condition is particularly useful for modeling most types of inlets. There are many ways to implement this boundary condition. Some applications, such as the NASA flow solvers CFL3D,[12] FUN3D,[13] and OVERFLOW,[14] simply set the back pressure at the boundary and then extrapolate the other fluid state quantities from the domain interior. Alternatively, the approach below uses the Riemann invariants to compute the boundary fluxes, similar to the strategy of Allmaras.[15]

Formulation

In the implementation of this boundary condition, the solver assumes the pressure at the boundary is a constant value (p_{set}) chosen by the user. All other flow attributes are determined by the flow leaving the computational domain through the boundary. Consider subsonic flow leaving the domain interior through the boundary as shown in Fig. 1. Using the Riemann invariants (J^+, J^-), the state at the boundary is first computed.

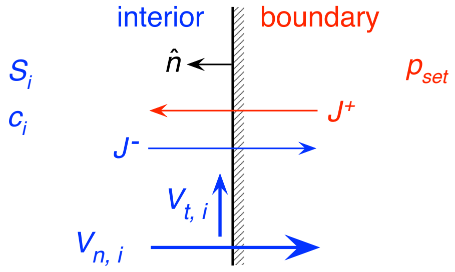

Figure 1. Back pressure boundary condition for subsonic outflow.

Provided that the flow is isentropic along characteristics, entropy can be extapolated from the domain interior (i) to the boundary (b).

$$S_b = S_i = p_i / \rho_i^{\gamma} \tag{3}$$

The Riemann invariant leaving the domain is computed using the flow quantities from the interior.

$$J^- = -V_{n,i} + \frac{2c_i}{\gamma - 1} \tag{4}$$

The speed of sound at the boundary is related to the pressure (set by the user) and entropy at the boundary.

$$c_b^{\,2} = \gamma p_{set}^{\frac{\gamma-1}{\gamma}} S_b^{\frac{1}{\gamma}} \tag{5}$$

The speed of sound at the boundary is also related to the two invariants.

$$c_b = \frac{\gamma-1}{4}(J^- - J^+) \tag{6}$$

Using the value of the exiting invariant and the user-specified back pressure, the invariant which enters the domain from the boundary can be computed.

$$J^+ = J^- - \frac{4}{\gamma-1}\sqrt{\gamma p_{set}^{\frac{\gamma-1}{\gamma}} S_b^{\frac{1}{\gamma}}} \tag{7}$$

Using the speed of sound and the entropy at the boundary, the density at the boundary can be determined.

$$\rho_b = \left(\frac{c_b^{\,2}}{\gamma S_b}\right)^{\frac{1}{\gamma-1}} \tag{8}$$

Also, the velocity normal to the boundary is computed using the invariants.

$$V_{n,b} = -\frac{1}{2}(J^- + J^+) \tag{9}$$

The remaining velocity components, which are parallel to the boundary surface, simply pass through from the interior.

$$V_{t,b} = V_{t,i} \tag{10}$$

At this point, the entire flow state at the boundary is available, which is then used to directly compute the flux density tensor at the boundary, $\overline{\overline{F}}_b$.

Safeguards

This boundary condition is only valid if the flow is indeed outflow. For a back pressure value that is too high, the boundary condition will actually try to induce a back flow, resulting in the boundary transitioning to inflow and violating the boundary condition requirements. As a safeguard in the implementation, the boundary condition was implemented so that it reverts to a solid wall boundary should the fluid try to flow back into the domain. When this occurs, only the pressure is required at the boundary, which is taken from the interior domain. The boundary condition is also only valid if the flow speed in the direction normal to the boundary is subsonic. If the back pressure is too low and the flow in the domain becomes supersonic, the boundary condition reverts to simple extrapolation, where the entire flow state in the interior is used to compute all fluxes through the boundary surface. With these two safeguards, the boundary condition algorithm can robustly handle either invalid user-provided data and/or flow transients during the solution process.

One other issue can occur either when the flow field is initialized to supersonic conditions or transients drive the flow near the boundary to become supersonic. In both of these cases, the supersonic extrapolation safeguard could prevent the flow from ever slowing to subsonic at the boundary, no matter what back pressure value is set by the user. To prevent this "lockout" situation, the Rankine-Hugoniot normal shock flow relations are exploited. The supersonic interior flow state is used to compute the pressure behind a normal shock right at the boundary. If the back pressure value provided by the user is higher than the post-shock pressure, the normal shock is assumed to actually exist at the boundary. The post-shock flow state is then used at the boundary to compute the flux density tensor. As the flow field evolves, this transient state forces subsonic flow to occur at the boundary, thus allowing the algorithm to proceed normally and not invoke the supersonic safeguard. However, if the pressure provided by the user is lower than the post-shock value, then the flow will continue to exit the boundary as supersonic flow, which is of course the correct choice when the back pressure is sufficiently low.

B. Constant Velocity Subsonic Outflow

Many popular flow solvers exclusively provide the back pressure boundary condition described above to model domain outflow within a duct. Often a user wishes to impose a specific mass flow rate through an inlet boundary surface. The user will then iterate on a back pressure value to obtain that mass flow rate. This iteration process is often computationally expensive and not always robust when strong nonlinearities exist in the flow field. In the end, the fact that the pressure is set to be constant over the boundary is not always necessary to the desired simulation result; just attaining the correct mass flow rate might be sufficient.

The back pressure boundary condition is often used on a surface that is actually a surrogate for a fan or compressor face, such as that inside the inlet of an air-breathing engine. However, constant pressure at the fan face is not always realistic in a duct without a long extent of constant cross-sectional shape ahead of the boundary surface. This was postulated by Pearson[16] and observed experimentally by Reid.[17] These authors suggest that outside the boundary layer, it is in fact the velocity that remains closer to constant over the fan face than pressure, particularly in a duct that is still varying in shape near the boundary surface. Inspired by this observation, a subsonic outflow boundary condition that assumes constant normal velocity (similar to that suggested by Gustafsson[18] and Yee[9]) was developed for Cart3D.

Note that this type of boundary condition is really only applicable to subsonic inviscid outflow. Applying this type of boundary condition to a duct using a viscous solver would not be appropriate since, without modification, it would not preserve the boundary layer velocity profile of the flow exiting the domain.

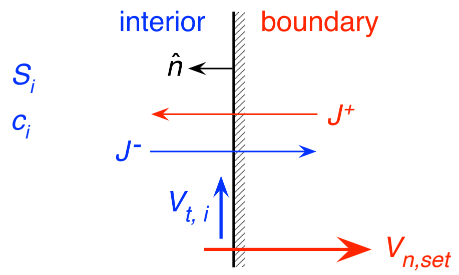

Figure 2. Constant normal velocity boundary condition for subsonic outflow.

Formulation

The constant outflow velocity boundary condition is shown schematically in Fig. 2. Similar to the back pressure boundary condition, Riemann invariants along characteristics are used in this algorithm. The process begins by extrapolating the domain entropy using Eq. (3) to the boundary. The exiting Riemann invariant is computed using Eq. (4). Using the user-specified normal velocity, the entering invariant is then computed.

$$J^+ = 2V_{n,set} - J^- \tag{11}$$

The speed of sound at the boundary is then computed using Eq. (6) and the density with Eq. (8). The pressure of the boundary state is computed using the definition of the speed of sound.

$$p_b = \frac{\rho_b c_b^2}{\gamma} \tag{12}$$

As with the back pressure boundary condition, the tangential velocity components at the boundary are extrapolated from the interior per Eq. (10). Using these boundary values, the flux density tensor is computed directly in the flow solver.

Safeguards

As with the back pressure boundary condition, this constant velocity condition is only valid if the outflow is subsonic. Two safeguards are implemented to ensure the algorithm remains valid. If the interior flow is subsonic and the computed boundary outflow Mach number is supersonic, the boundary state is recomputed to be sonic and thus simulate choked flow. The second safeguard allows supersonic flow to simply exit the domain properly. If the interior flow is supersonic and the specified outflow is also supersonic, the interior state is used as the boundary state and the fluxes computed appropriately using simple extrapolation of all interior quantities.

An advantage to using this boundary condition as opposed to the more common constant pressure condition is that controlling mass flow rate through the boundary is much more straightforward. Instead of iterating on the back pressure, the solver can simply set the exit velocity based on the current value of the area-weighted integral of density. Density usually does not vary significantly over the boundary and therefore the iteration process is quite robust. More on this mass-flow-rate steering is discussed in Section E.

C. Stagnation State Subsonic Inflow

While the previous boundary conditions discussed are applicable to inlets in a propulsion system, the next two are useful to model flow into the computational domain from a combustor, turbine, or nozzle plenum. For a valid subsonic inflow boundary condition, four flow quantities are set by the boundary state and only one quantity is taken from the interior flow. A commonly used inflow boundary condition is one where stagnation quantities are set by the user. More specifically, the stagnation pressure and stagnation temperature along with the inflow direction are specified.

Formulation

This boundary condition implementation is identical to that in FUN3D[13] and is represented in Fig. 3. The inflow direction at the boundary is assumed to be normal to the surface, which sets the other two flow quantities. The boundary condition assumes the flow crossing the boundary is adiabatic and therefore extrapolates the enthalpy from the interior flow.

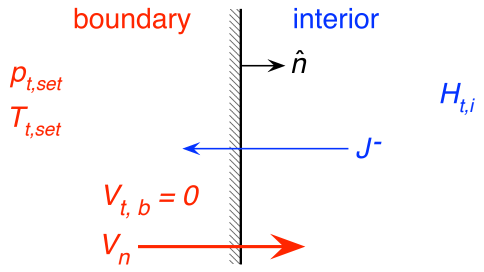

Figure 3. Stagnation property boundary condition for subsonic inflow.

The first step of the algorithm is to compute the total enthalpy,

$$H_{t,i} = \frac{c_i^2}{\gamma - 1} + \frac{1}{2}V^2 \tag{13}$$

where $V^2 = u_i^2 + v_i^2 + w_i^2$, and the value of the Riemann invariant running upstream from the domain interior.

$$J^- = -V + \frac{2c_i}{\gamma - 1} \tag{14}$$

Recall that the flow direction at the boundary is forced to be normal to the boundary surface, and thus the tangent velocity components are both set to zero.

$$V_{t,b} = 0 \tag{15}$$

This means the flow velocity normal to the boundary is indeed the total velocity. Extrapolating Eqs. (13) and (14) to the boundary and combining them, the speed of sound at the boundary is the largest root of the following quadratic equation.

$$H_{t,i} = \frac{c_b^2}{\gamma - 1} + \frac{1}{2}\left(J^- - \frac{2c_b}{\gamma - 1}\right)^2 \tag{16}$$

Using the speed of sound and the invariant, the velocity and hence the Mach number at the boundary are then computed. In the end, this amounts to one flow quantity that is taken from the interior flow.

$$V_{n,b} = \frac{2c_b}{\gamma - 1} - J^- \quad M_b = \frac{V_{n,b}}{c_b} \tag{17}$$

Using the Mach number and the user-provided values of total pressure and temperature, the static pressure and temperature can be computed at the boundary.

$$p_b = p_{t,set}\left(1 + \frac{\gamma - 1}{2}M_b^2\right)^{\frac{-\gamma}{\gamma - 1}} \tag{18}$$

$$T_b = T_{t,set}\left(1 + \frac{\gamma - 1}{2}M_b^2\right)^{-1} \tag{19}$$

Finally, using the perfect gas law, the density at the boundary is computed.

$$\rho_b = \frac{p_b}{RT_b} \tag{20}$$

At this point, the entire boundary state is known and the flux density tensor can be directly computed.

Safeguards

This boundary condition is only valid if the inflow is subsonic. Hence, a safeguard is implemented that restricts the Mach number at the boundary to be no higher than sonic. To specify supersonic inflow the original boundary condition implemented in Reference 2 would be used. If the user specifies values of total pressure and temperature that are too low to produce inflow, another safeguard is implemented to revert the boundary condition to a solid wall treatment.

D. Constant Mass-Flow-Rate Subsonic Inflow

The final boundary condition handles subsonic inflow with specified mass flow rate and total temperature. The user provides these two flow attributes and, as before, the solver assumes the inflow direction at the boundary is normal to the surface, thus providing the other two flow attributes. This boundary condition is represented in Fig. 4.

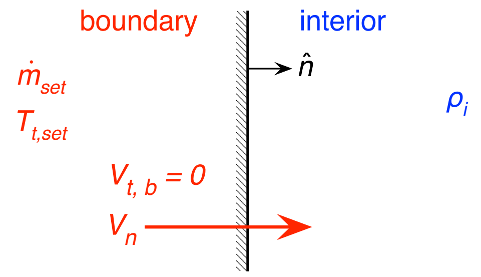

Figure 4. Specified mass flow rate boundary condition for subsonic inflow.

Formulation

This boundary condition is simple. First, the density at the boundary is extrapolated downstream from the interior.

$$\rho_b = \rho_i \tag{21}$$

Equation (15) again applies because of the tangential velocity assumption. The mass flow rate through the boundary is given, so the flow velocity normal to the boundary (which is also the total velocity) can be directly computed.

$$V_n = V = \frac{\dot{m}_{set}}{A_b \rho_b} \tag{22}$$

Using the energy equation and the provided total temperature, the temperature at the boundary can be computed.

$$T_b = T_{t,set} - \frac{\gamma - 1}{2\gamma R} V^2 \tag{23}$$

This allows the pressure of the flow state at the boundary to be computed using the perfect gas law given in Eq. (20). At this point the complete boundary state is known and the flux density tensor at the boundary can be directly computed.

Safeguards

As with the stagnation state boundary condition (Section C), this boundary condition is only valid if the inflow is subsonic. Hence, a safeguard is implemented that forces the Mach number at the boundary to be no higher than sonic. Should the user desire supersonic inflow, then simple extrapolation from a user-provided state would be used.[2]

E. Mass-Flow-Rate Steering

As was mentioned in Section B, the constant velocity outflow boundary condition is quite amenable to mass-flow-rate control or steering. It is very common for the user to know the mass flow rate through an inlet but not know the corresponding value of constant pressure at the fan-face boundary. Hence the ability to robustly and quickly steer a boundary condition to produce a particular mass flow rate is very desirable.

Mass flow rate through a boundary surface is given by

$$\dot{m} = \rho V_n A_b \tag{24}$$

The boundary surface area is constant and the density is often nearly constant over the boundary surface, especially inside an inlet. Thus, mass flow rate and velocity are tightly linked.

Mass flow rate steering has been implemented in Cart3D to work with the constant velocity boundary condition. The user specifies the total mass flow rate, a convergence tolerance, and the number of relaxation steps to run between updates of the outflow velocity at the boundary. To be more specific, the algorithm is as follows:

1. Run flow solver for a user-specified number of startup relaxation steps
2. Integrate density over the boundary surface area:

$$(\rho A)_b = \int_{A_b} \rho_b dA_b \tag{25}$$

3. Compute current value of mass flow rate through boundary:

$$\dot{m} = (\rho A)_b V_{n,set} \tag{26}$$

4. Compare error in actual mass flow rate (difference between current and desired) to user-specified tolerance. If within tolerance, do nothing to the boundary condition values and continue converging Euler solver. If outside tolerance, set outflow velocity at boundary to a new value:

$$V_{n,set} = \frac{\dot{m}_{set}}{(\rho A)_b} \tag{27}$$

5. Run user-specified number of iterations (update interval). Go to step 2 and continue.

To ensure stability of the convergence of this algorithm in practice, some under-relaxation is applied when computing a new value of the boundary outflow velocity. Using this algorithm, the mass flow rate through the outflow boundary surface can be steered robustly to the user-specified value.

To control the mass flow rate through an inflow boundary, the constant mass-flow-rate boundary condition (Section D) is obviously appropriate. In this case, the mass flow rate does not need to be steered since it is directly set in the boundary condition itself.

F. Adjoint Implmentation

The boundary conditions described in Sections A-D have been incorporated into an existing mesh adaptation framework that is based on the method of adjoint weighted residuals.[19] The implementation turned out to be relatively straightforward — the boundary conditions were linearized and included in the transpose of the flow Jacobian matrix. Note, however, that this involved the linearization of the safeguards. In practice, we find that the flow solutions on coarse meshes or for conceptual design geometries frequently activate some of the safeguards, even at steady-state. As such occurrences should not terminate the mesh adaptation procedure, the safeguards were therefore linearized. In addition, we have studied the behavior of the adjoint solution near the boundary faces. The adjoint variables in these regions appear smooth and continuous. This is important since the error estimation procedure uses interpolation operators that assume smoothness of the adjoint field. An important factor for preserving duality in our implementation is that the boundary flux density tensor is computed directly from the boundary state in the flow solver without using a Riemann solver as is done by Pandya.[2]

III. Applications

To demonstrate the use of the boundary conditions detailed in Section II along with the original propulsion boundary condition implemented in Cart3D, several generic propulsions systems were analyzed. These examples include a ducted fan in hover, a turbofan in transonic flow, a turbojet with a two-dimensional inlet in supersonic flow, and a scramjet in hypersonic flow. For a less academic example, the boundary conditions were also applied to a modern low boom supersonic aircraft design with multiple boundaries of inflow and outflow. All solutions were generated using adjoint-based adaptive mesh refinement.

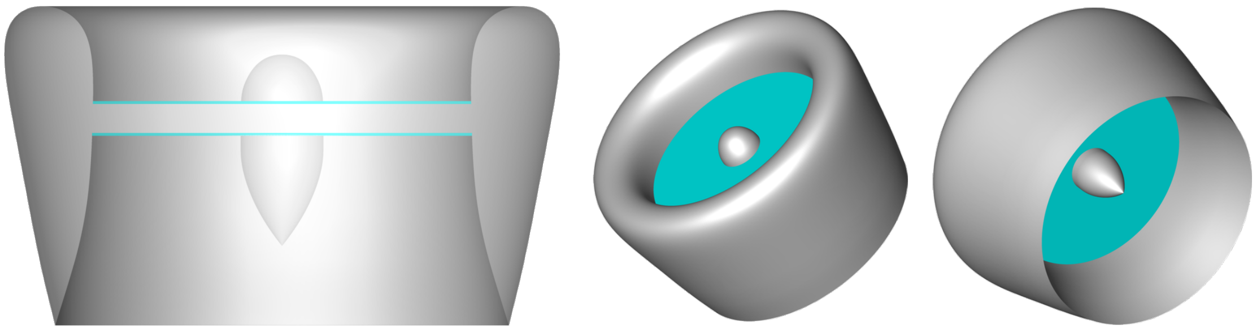

Figure 5. Notional ducted fan geometry. Cyan faces represent the propulsor disk where propulsion inflow/outflow boundary conditions are applied.

A. Ducted Fan in Hover

The first example presented to demonstrate the boundary condition implementation is a model of a ducted fan. The model geometry is depicted in Fig. 5. The fan itself is modeled as a simple propulsor disk of finite thickness (shaded in cyan) by applying the appropriate boundary conditions on each side of the disk. The freestream Mach number is very low ($M_\infty = 0.001$) and no angle of attack was imposed inducing axisymmetric flow. Therefore this case represents the fan in a near-hover (at worst a slow climb) state in essentially incompressible flow.

For all simulations on this geometry, the output functional of interest is total drag (negative values imply thrust). The adaptive mesh refinement procedure therefore tailors the mesh to minimize discretization errors in drag. All meshes used a minimum farfield distance of at least 30 fan diameters with an extended farfield below the duct. This farfield distance was refined 20 levels so the length of the smallest cell edges in the finest adapted mesh were just over 0.006% of the fan diameter. All final meshes consisted of over 100 million cells. Note that while mesh-converged solutions occurred with significantly smaller meshes, the larger meshes were used to fully test the boundary conditions with all mesh sizes.

The ducted fan was first run with the back pressure outflow and stagnation property inflow boundary conditions. The boundary condition states were selected to generate some thrust but matching of the mass flow rate through both faces was not enforced in this simulation. Hence, while this case does demonstrate these two boundary conditions, the solution itself does not represent a realistic case. To match mass flow rates using these boundary conditions would take some iteration of both boundary conditions. For such a low speed case where the entire flow field is highly sensitive to any changes anywhere on the surface or through the propulsor disk, this procedure could be computationally expensive.

A qualitative assessment of the solution can be made by examining the Mach number and normalized pressure contours on a symmetry plane of the flow field depicted in Fig. 6. Both sets of contours are smooth in the region of the fan on both sides suggesting a proper implementation. Note the pressure is essentially constant just above the propulsor disk where the back pressure outflow boundary condition is applied. This pressure value also matches the value specified for the boundary condition in this simulation, further implying a proper implementation.

A more quantitative assessment of the outflow boundary condition can be made by examining the pressure distribution interpolated from the Cartesian mesh onto the propulsor disk surface. The difference in that interpolated pressure from the value used in the boundary state provides an estimate of discretization error at the boundary face. If the boundary condition is implemented correctly, this error should vanish as the mesh is refined. Figure 7 shows contours of this error (as a percentage of the back pressure) for solutions on an intermediate and the final mesh. The L_2 norm of this error distribution for all meshes used throughout the adjoint-driven mesh refinement process is shown in Fig. 8. The overall decrease in this error norm, particularly on the finest meshes, asserts a proper implementation of the boundary condition.

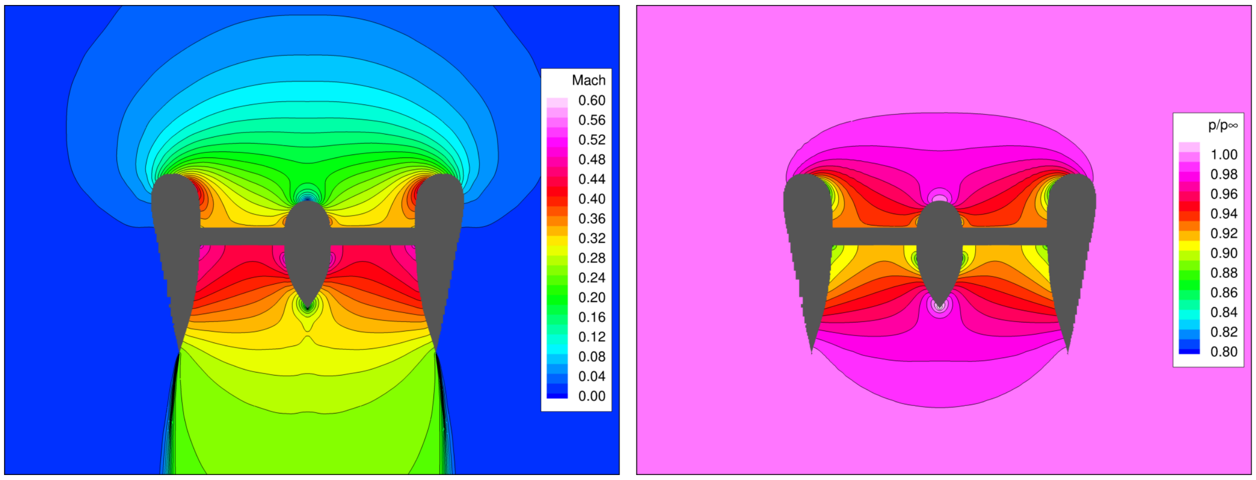

Figure 6. Mach number and pressure contours of flow through a ducted fan demonstrating the back pressure outflow and stagnation property inflow boundary conditions. $M_\infty = 0.001$, 130 million cells.

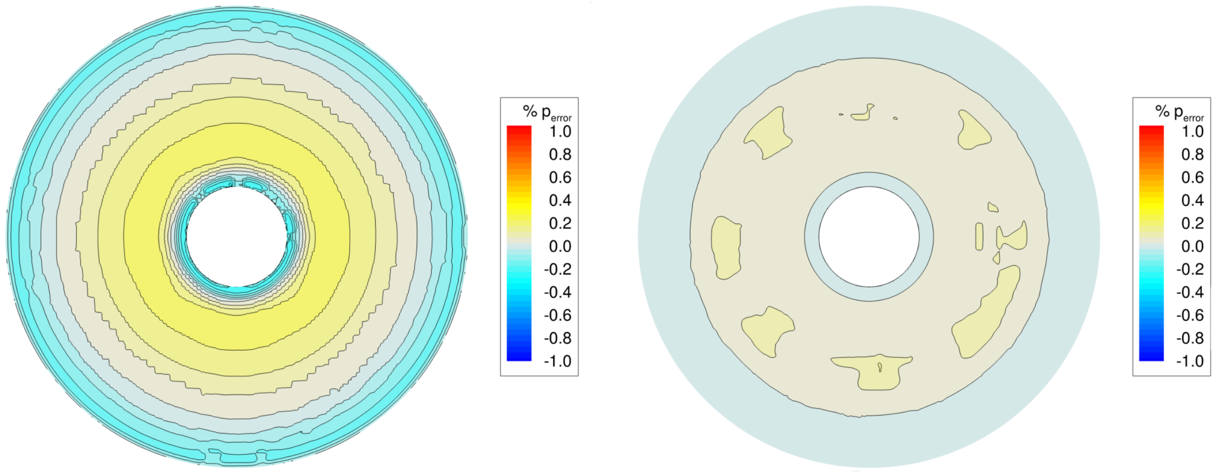

Figure 7. Contours of percent error in pressure from set boundary condition value on outflow face. Left image is from the mesh with 2 million cells, while the right image mesh has 130 million cells.

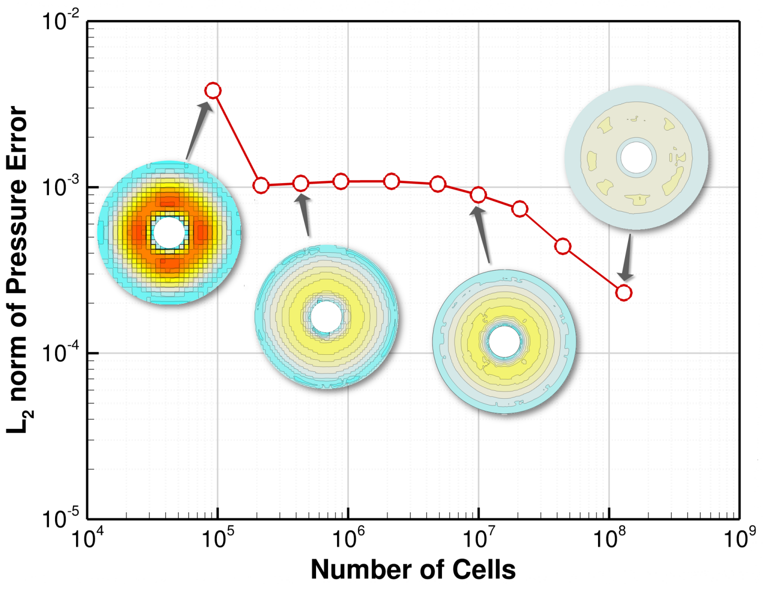

Figure 8. Mesh convergence of L_2 norm of pressure error at outflow boundary of ducted fan. Red and blue contours represent higher values of error while light grey contours are closer to zero error.

A similar solution was generated using the other two newly implemented boundary conditions, namely constant velocity outflow and constant mass flow rate inflow. Again, the mass flow rates into and out from the propulsor disk were not matched. The flow contours on this solution are presented in Fig. 9. Discounting the different mass flow rates, note the subtle differences in solution contour shapes near the propulsor disk between the two sets of applied boundary conditions (Figs. 6 and 9). On an actual geometry, the user would have to choose the boundary condition that more closely models reality if the solution accuracy nearest the propulsor disk is critical. The error distribution in the mass flow rate on the inflow side of the disk is shown in Fig. 10 for both an intermediate mesh and the final mesh. The mesh convergence of the L_2 norm of this error is shown in Fig. 11. The steady decrease in error as the mesh is refined indicates a proper implementation of the constant mass-flow-rate inflow boundary condition. Note that the errors in total temperature at the boundary follow a similar rate of convergence to the correct value.

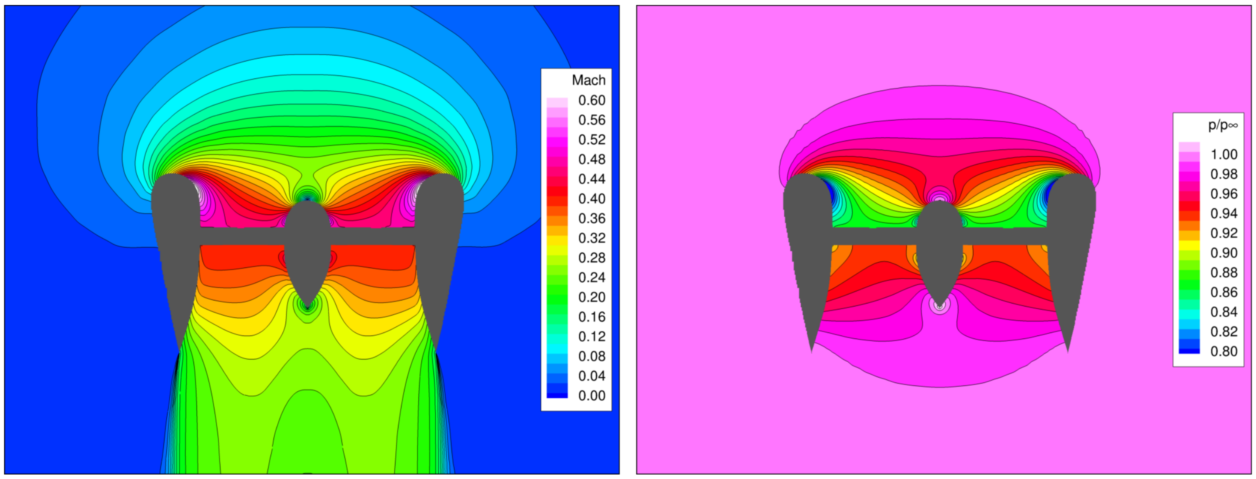

Figure 9. Mach number and pressure contours of flow through ducted fan demonstrating the constant velocity outflow and constant mass-flow-rate inflow boundary conditions. M_∞ = 0.001, 160 million cells.

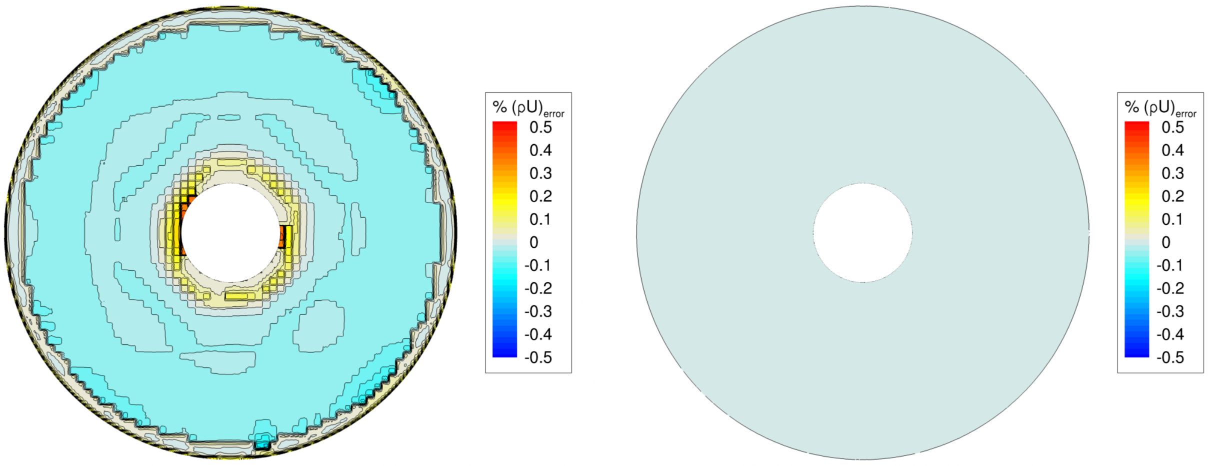

Figure 10. Contours of percent error in mass flow rate from set boundary condition value on inflow face. Left image is from the mesh with 0.5 million cells, while the right image mesh has 160 million cells.

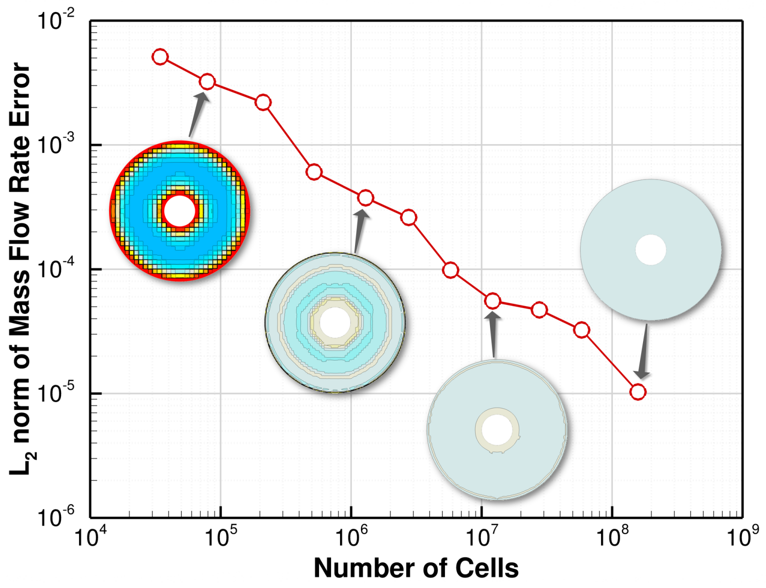

Figure 11. Mesh convergence of L_2 norm of mass-flow-rate error at inflow boundary of ducted fan. Red and blue contours represent higher values of error while light grey contours are closer to zero error.

A final simulation was run where the mass flow rate through the inflow and outflow surfaces were forced to match. The constant mass-flow-rate boundary condition was used in the inflow region, and the constant velocity outflow boundary condition was steered to match this mass flow rate using the process described by Eqs. (25)-(27). The initial guess for the outflow velocity was intentionally initialized to be very incorrect (> 30% error) for matching the mass flow rate below the disk. Figure 12 shows the convergence of the mass flow rate through the outflow boundary along with the functional (drag) over the 10 adaptation cycles of the simulation. Each circle in the plot represents a mass-flow-rate update (every 20 iterations) and there are at least 100 iterations in each adaptation cycle. If the mass-flow-rate error is greater than the user-specified tolerance (< 0.003% for this case), the outflow velocity is updated using Eq. (27). The discontinuities in both plotted quantities in Fig. 12 occur where the mesh is adaptively refined. The mass-flow-rate steering begins at iteration 500 and quickly converges close to the desired value in just a few iterations, even before reaching the next mesh adaptation cycle. Small errors in the mass flow rate are reintroduced every time the mesh is refined, but the steering algorithm quickly adjusts the boundary condition to reacquire the desired mass flow rate though the outflow surface.

Note that since the flow is nearly incompressible in this case, mass flow rate matching could have easily been accomplished for this case without the steering algorithm as long as the same boundary conditions were used. Constant density throughout the flow field implies the constant velocity outflow boundary condition allows the user to implicitly specify the mass flow rate. On the other side of the disk, the inflow boundary condition explicitly sets the mass flow rate below the disk. Hence, mass flow rate matching would be very easy with these boundary conditions. Nonetheless, for more general flows where the density varies greatly, the mass flow steering algorithm is still quite valuable.

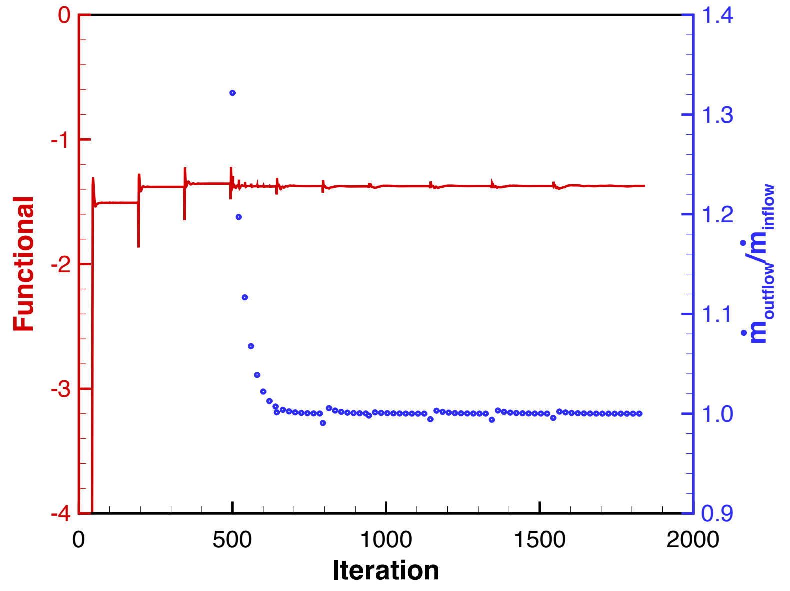

Figure 12. Mesh convergence of mass flow rate and functional (drag) on ducted fan in hover.

The convergence of the functional and the adjoint-based error indicator for each cycle of grid refinement is shown in Fig. 13. Note the steady reduction in the error estimate and the converging functional value. The final mesh generated for this last case on the ducted fan using adjoint-driven adaptive mesh refinement is shown in Fig. 14. It consisted of over 180 million cells. As expected, the highest levels of mesh refinement occurred mostly near the duct and propulsor disk surfaces and in the shear layer generated by the stream of flow exiting the duct. The mesh is also more refined where the attachment line of flow on the outer duct occurs, which is near the trailing edge. Mach number and pressure contours generated using this adapted mesh are presented in Fig. 15. Note that the Mach number of the flow above and below the disk is nearly identical. For this nearly incompressible flow where the speed of sound and density are essentially constant everywhere, this also means the velocity on both sides of the disk are nearly the same, asserting matched mass flow rates.

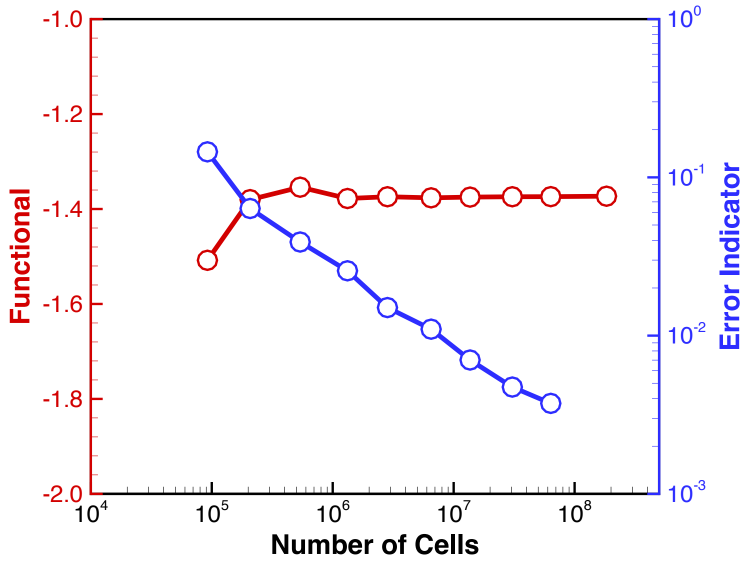

Figure 13. Mesh convergence of functional (drag) and error estimate on ducted fan.

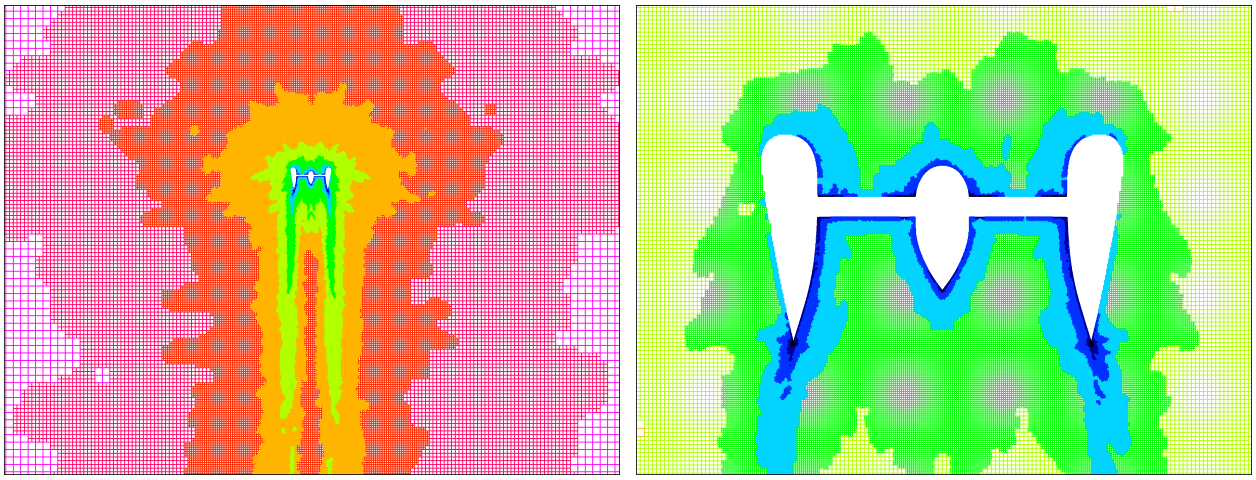

Figure 14. Adjoint-driven refined mesh (180 million cells) for ducted fan with steered outflow rate to match the inflow rate. Colors represent cells of similar levels of refinement.

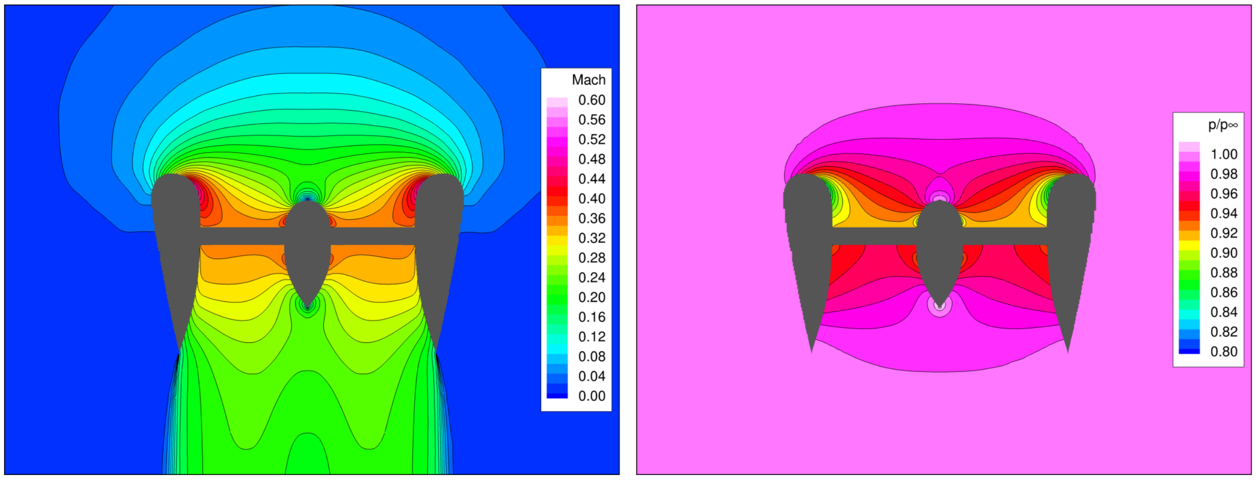

Figure 15. Mach number and pressure contours of flow through ducted fan demonstrating steered outflow rate to match the inflow rate. $M_\infty = 0.001$, 180 million cells.

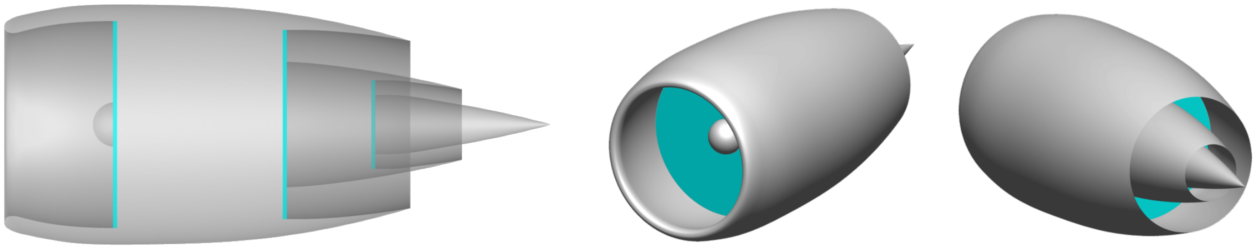

Figure 16. Notional two-stream turbofan geometry. Cyan faces represent where propulsion inflow/outflow boundary conditions are applied.

B. Transonic Nacelle

The second generic propulsion system considered is a turbofan in transonic flow with both fan and turbine exhaust flows. The geometry model is shown in Fig. 16. The surfaces shaded in cyan indicate where propulsion inflow/outflow boundary conditions are applied. These include the annulus just in front of the notional fan face (outflow), the annulus just beyond the fan (inflow), and the annulus just past the turbine exit (inflow). In other words, the details of the flow paths through the fan, compressor, burner, and turbine are not modeled.

The turbofan geometry was run in transonic flow (Mach 0.8) at zero angle of attack. Two different cases were run with the same combinations of the new boundary conditions used on the ducted fan. Neither case enforced matching mass flow rates. The functional used to adapt the mesh was total drag. The mesh farfield distance was roughly 40 fan diameters. All final adapted meshes were generated through 19 levels of refinement and consisted of about 160-170 million cells with the length of the smallest cell edges being just over 0.005% of the fan diameter. The Mach number and pressure contours on a symmetry plane for these two solutions are depicted in Figs. 17 and 18. Qualitatively, the solutions suggest the boundary conditions are properly implemented.

Contours of percent error in the total pressure on the inflow faces are shown in Fig. 19 for solutions on an intermediate and the final mesh. The L_2 norm of this error distribution for all meshes used throughout the mesh adaptation process is shown in Fig. 20. Note that though not presented, contours and convergence of total temperature error are very similar. The overall decrease in this error norm, particularly on the finest meshes, assert a proper implementation of the boundary condition. The error contours in velocity for the outflow boundary where the constant velocity outflow boundary condition was applied are shown in Fig. 21. The L_2 norm of this error distribution as the mesh is refined is shown in Fig. 22. The results assert a proper implementation of this outflow boundary condition.

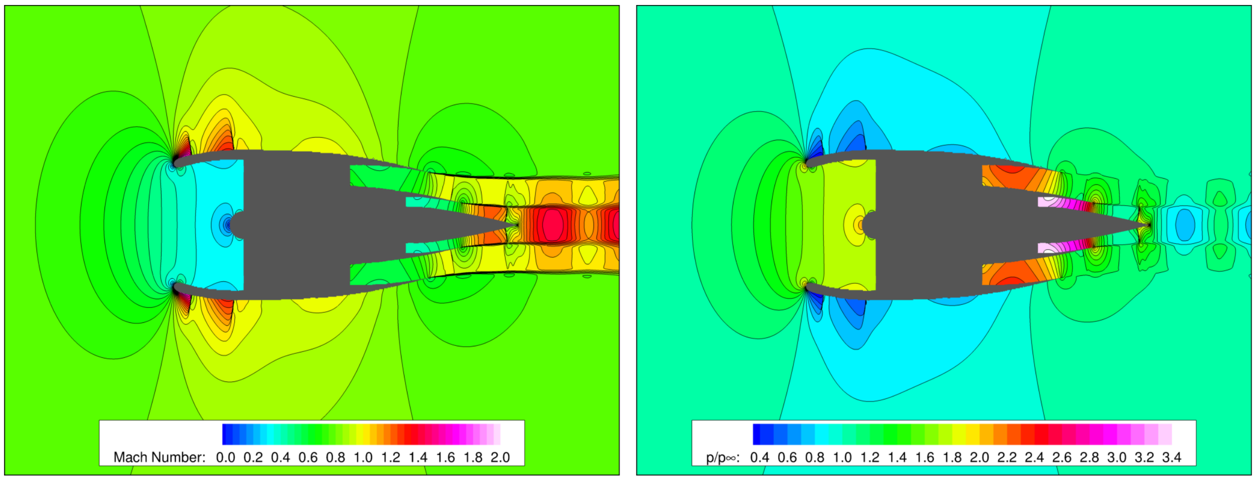

Figure 17. Mach number and pressure contours of flow through turbofan demonstrating the back pressure outflow and stagnation property inflow boundary conditions. $M_\infty = 0.8$, 170 million cells.

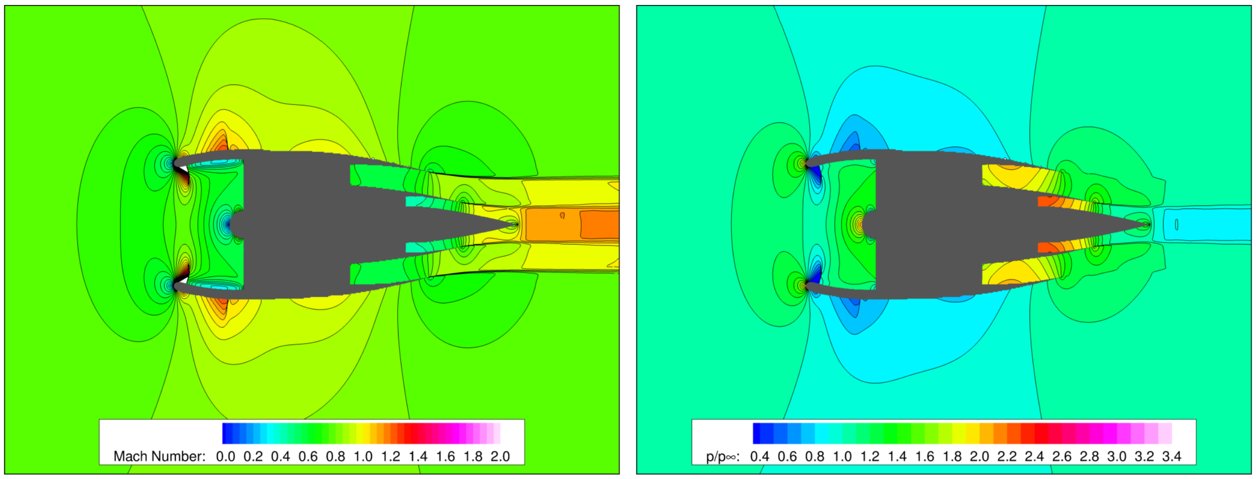

Figure 18. Mach number and pressure contours of flow through turbofan demonstrating the constant velocity outflow and constant mass-flow-rate inflow boundary conditions. $M_\infty = 0.8$, 160 million cells.

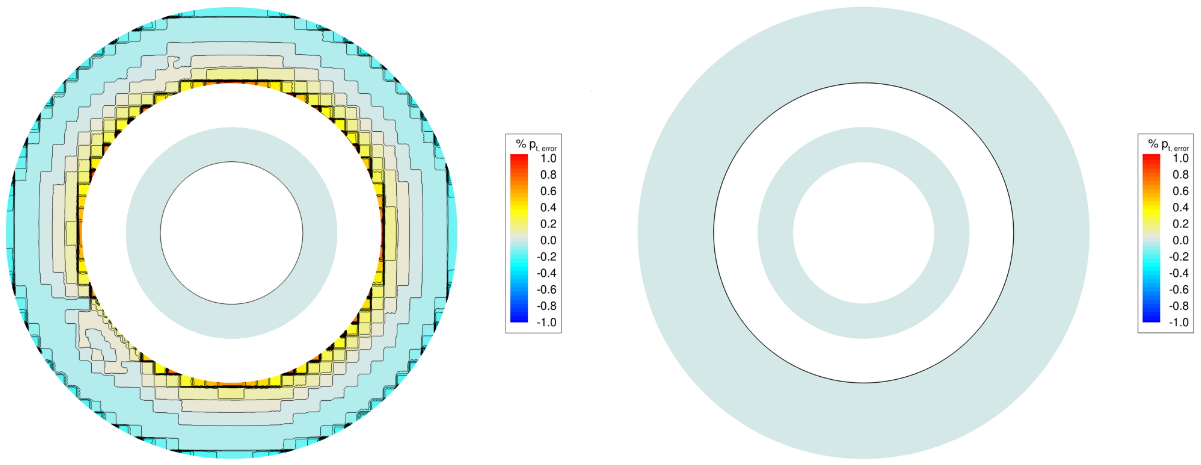

Figure 19. Contours of percent error in total pressure from set boundary condition value on inflow faces (fan and turbine exhaust). Left image is from the mesh with 0.2 million cells, while the right image mesh has 170 million cells.

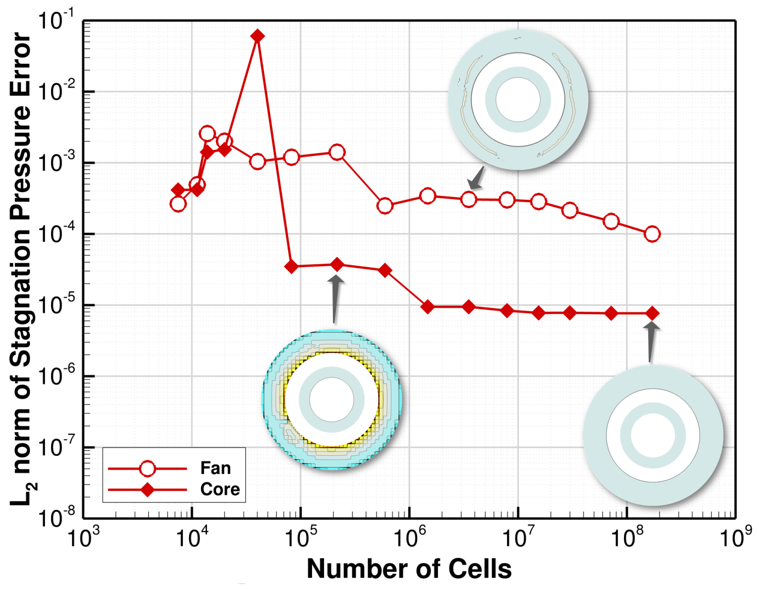

Figure 20. Mesh convergence of L_2 norm of total pressure error at inflow boundary of ducted fan. Red and blue contours represent higher values of error while grey contours are closer to zero error.

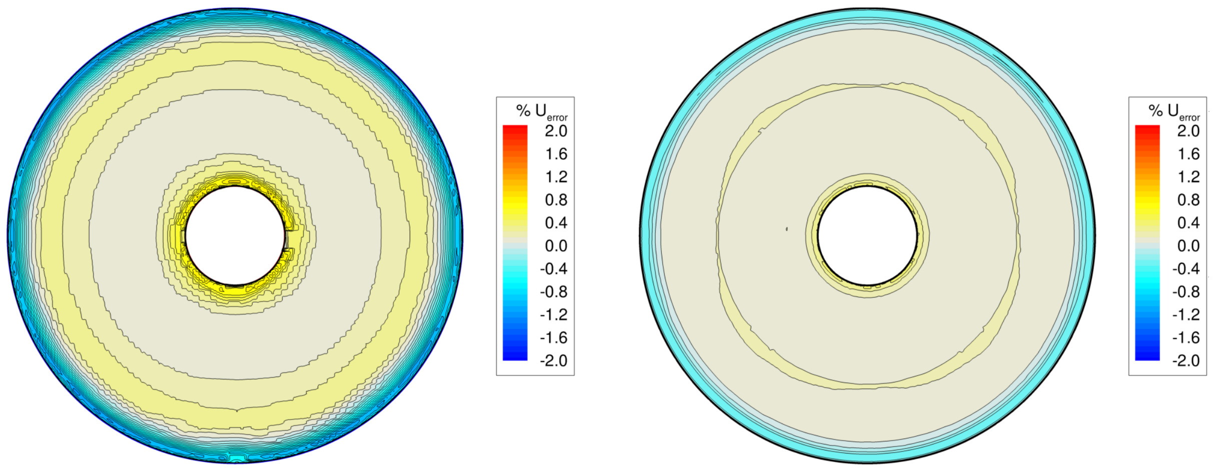

Figure 21. Contours of percent error in outflow velocity from set boundary condition value on outflow faces. Left image is from the mesh with 1.3 million cells, while the right image mesh has 160 million cells.

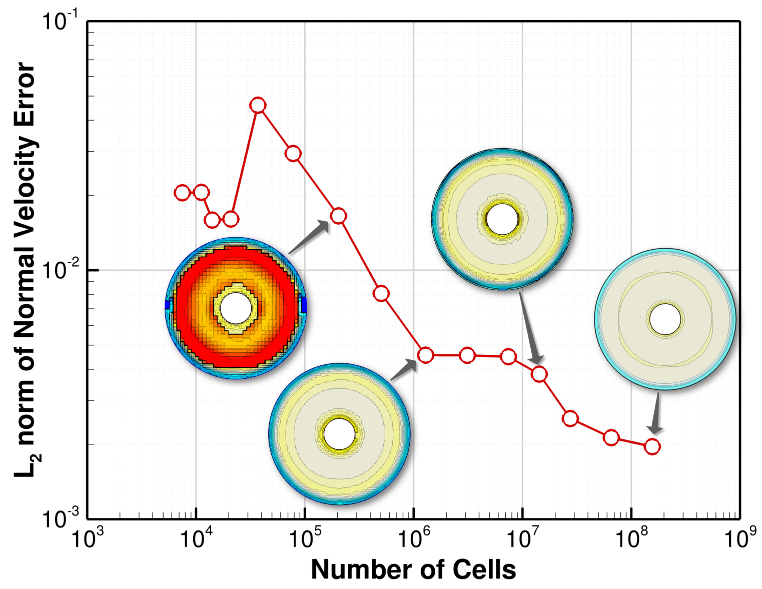

Figure 22. Mesh convergence of L_2 norm of outflow velocity error at outflow boundary of transonic nacelle. Red and blue contours represent higher values of error while grey contours are closer to zero error.

As was done with the ducted fan, a case was run where the exhaust flows used the constant mass flow rate boundary condition, and the inlet mass flow rate was steered to match the sum of the two streams. The original guess for the constant velocity boundary condition on the fan face was purposely chosen to be very different than the final value. The convergence of the inlet mass flow rate is shown in Fig. 23 along with the convergence of the functional (drag) over 13 adaptation cycles. Updates in the boundary condition were made every 20 iterations (each circle in the plot). The same behavior in the mass flow rate that was seen with the ducted fan is evident in this case as well. The mass flow rate converges very quickly for each adapted mesh, with only small jumps occurring when the mesh is refined. These jumps are quickly mitigated in just a handful of boundary condition updates.

The convergence of the functional and the corresponding error indicator for this case are shown in Fig. 24. In this case, we again see steady reduction in error and convergence of the drag functional. The final adapted mesh is shown in Fig. 25 and the solution is given in Fig. 26. Note the mesh is highly refined near the surface, at the edges of all propulsive stream tubes, and where any shocks are located.

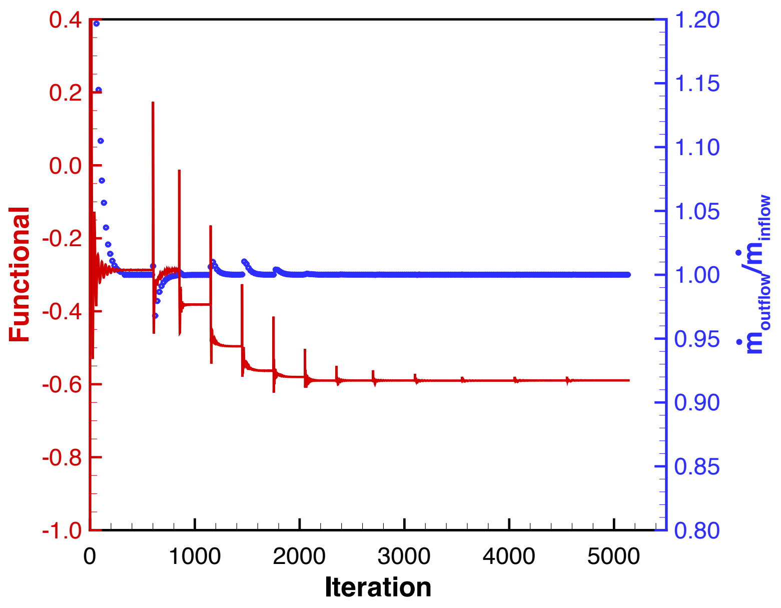

Figure 23. Convergence history of mass flow rate and functional (drag) on turbofan in transonic flow.

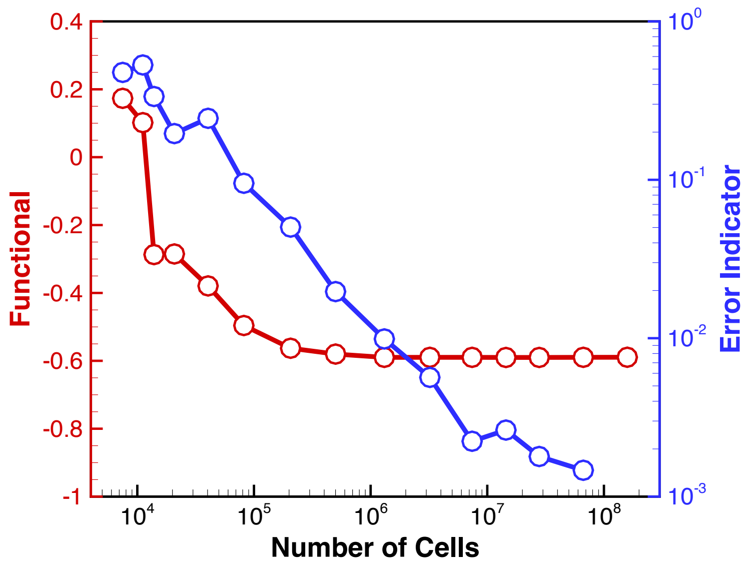

Figure 24. Mesh convergence of functional (drag) and error estimate on turbofan.

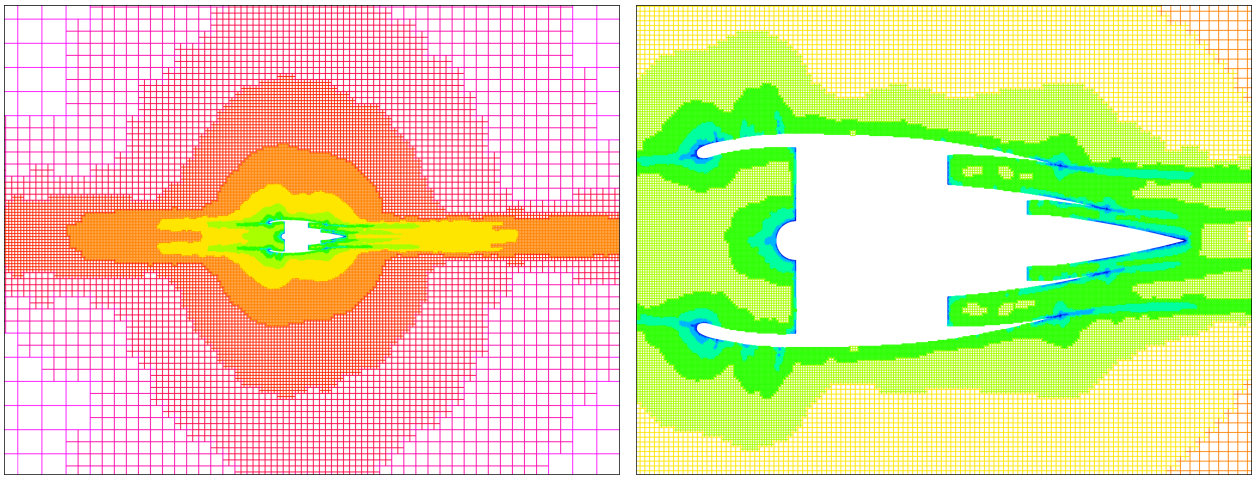

Figure 25. Adjoint-driven refined mesh (160 million cells) for turbofan with steered outflow rate to match the inflow rate. Colors represent cells of similar refinement.

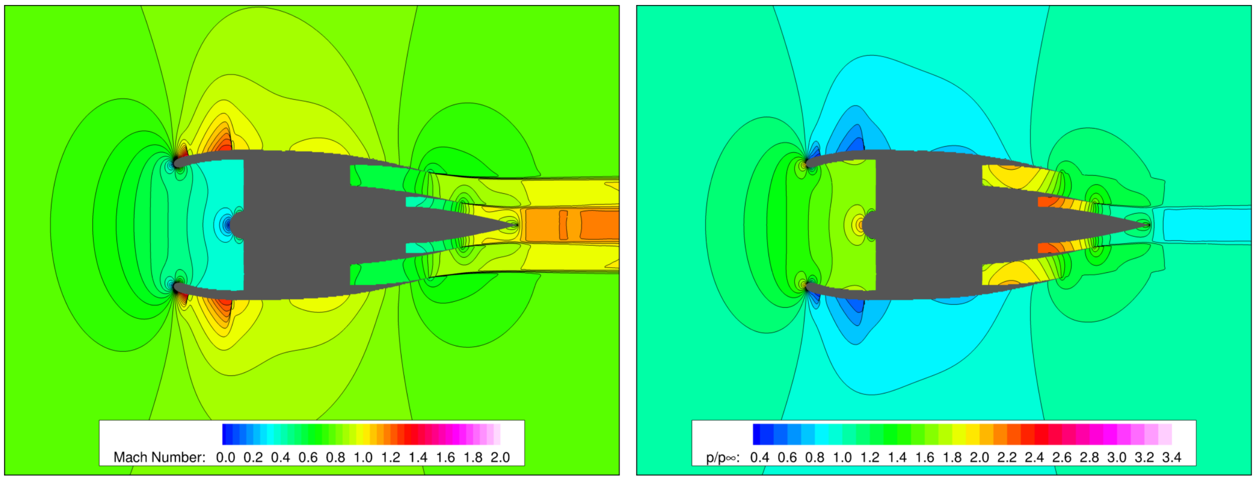

Figure 26. Mach number and pressure contours of flow through turbofan demonstrating steered outflow rate to match the inflow rate. M_∞ = 0.8, 160 million cells.

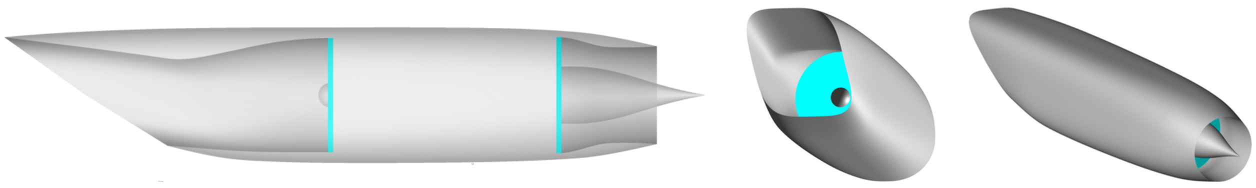

Figure 27. Notional turbojet with supersonic inlet geometry. Cyan faces represent where propulsion inflow/ outflow boundary conditions are applied.

C. Supersonic Nacelle

The third example case is an isolated turbojet nacelle in supersonic flow with a two-dimensional ramp inlet and converging-diverging nozzle with a center body. The geometry is shown in Fig. 27 where the cyan shaded surfaces represent where the propulsion inflow/outflow boundary conditions are applied. The inlet was designed to generate an external oblique shock along with a terminating normal shock which together act as the primary diffusion mechanism.

For these simulations, the freestream Mach number is 1.5 and angle of attack is 1.0°. Two solutions were generated with the same boundary condition combinations used in previous example cases. As in all other solutions, adaptive mesh refinement is used and here the functional used to drive refinement is an equally weighted sum of lift and thrust since the propulsion system is not axisymmetric. Final adapted meshes consisted of 19 levels of refinement and 120-170 million cells. The smallest cells were about 0.01% of the fan diameter in edge length. As with other cases, the compressor and turbine mass flow rates were purposely not matched.

The boundary conditions for these two cases were set to simulate off-design conditions to demonstrate the robustness of the formulation. Mach number and normalized pressure contours for these two cases are depicted in Figs. 28 and 29. In the first case where the back pressure outflow boundary condition is applied, the inlet is ingesting too much flow and the terminal normal shock has been swallowed by the inlet. In the case that used the constant velocity outflow boundary condition, the inlet has too much spillage, thus causing reverse flow right at the subsonic lip of the inlet and even some separation externally off this lip. Note that for both cases, the plume is not well defined in the contour plots. Since the flow is all supersonic from the nozzle back, the plume has no effect on the forces acting on the nacelle. Hence, the adaptive mesh refinement has ignored the plume region altogether leaving a very coarse mesh starting right behind the nozzle exit, resulting in a more economical mesh.

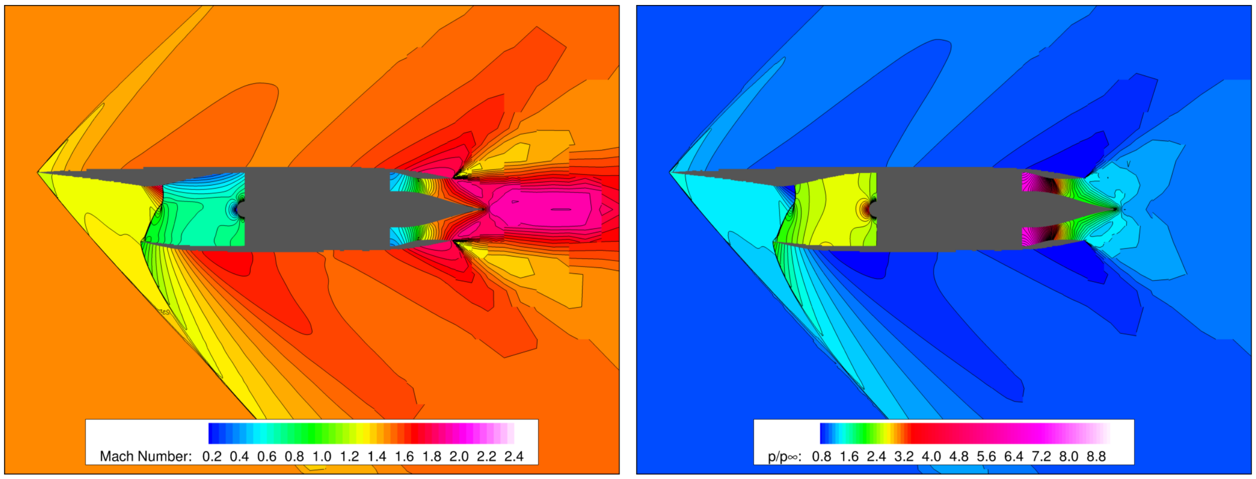

Figure 28. Mach and pressure contours of flow through turbojet demonstrating the back pressure outflow and stagnation property inflow boundary conditions. $M_\infty = 1.5$, 170 million cells.

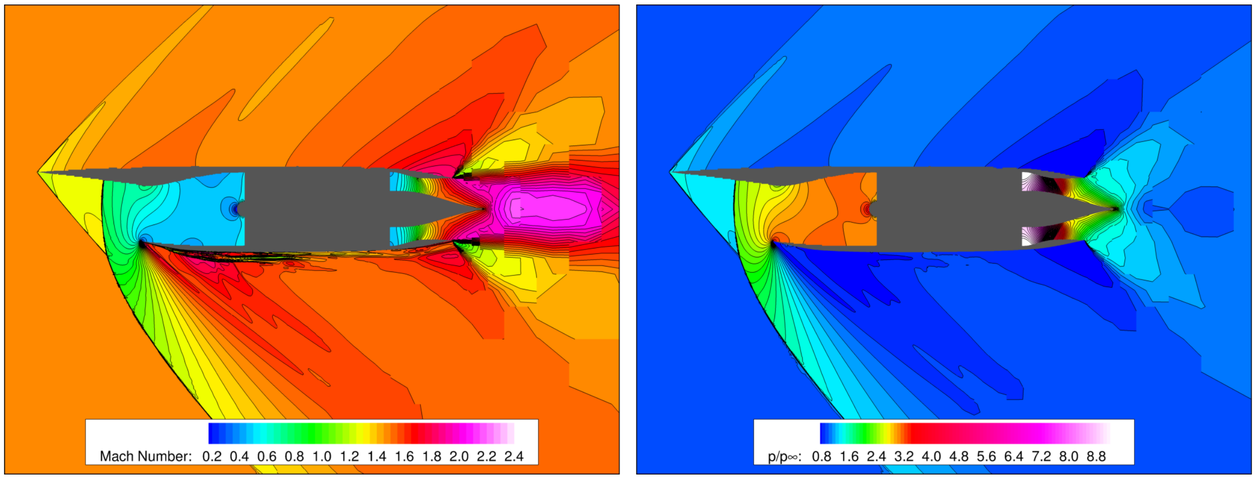

Figure 29. Mach and pressure contours of flow through turbojet demonstrating the constant velocity outflow and constant mass-flow-rate inflow boundary conditions. $M_\infty = 1.5$, 170 million cells.

A mass-flow-rate steering case was also run on this turbojet geometry. The convergence of that case is shown in Fig. 30. The outflow boundary condition was updated every 25 iterations. Note that even after the mesh was refined, the mass flow rate did not deviate much if at all. For this case, a pressure line sensor was also placed a short distance behind the nozzle. This pressure sensor is the same as those used by Aftosmis[20] to capture near-field pressure signatures for sonic boom calculations. Adding the sensor measurement to the functional forced the mesh refinement process to encourage resolution of the plume. Hence, for this case the functional used for mesh refinement was a weighted sum of lift, drag, and the square of the freestream-normalized pressure at the sensor. The convergence of this functional along with the error estimates at each mesh refinement is shown in Fig. 31. Note the functional converged quite well and the error estimate shows a steady reduction with each refinement of the mesh. The final refined mesh is shown in Fig. 32. As expected, the mesh is highly refined wherever a shock is present.

Mach number and pressure contours for this solution are shown in Fig. 33. Here the inlet is operating near the design point with the terminal shock just ahead of the subsonic lip (minimal spillage). Note the high resolution of the plume for this case as compared to the previous cases (Figs. 28 and 29) where no pressure sensor was included in the functional.

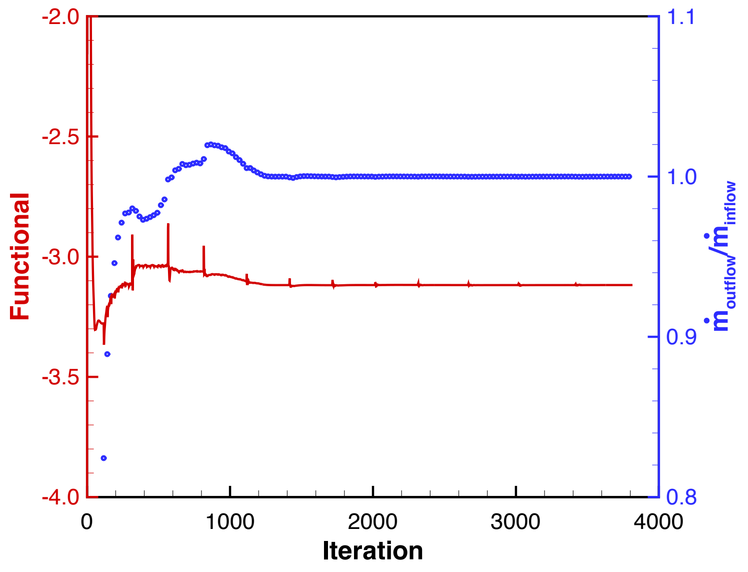

Figure 30. Convergence history of mass-flow-rate steering and functional (lift + drag + plume sensor) on turbojet in supersonic flow.

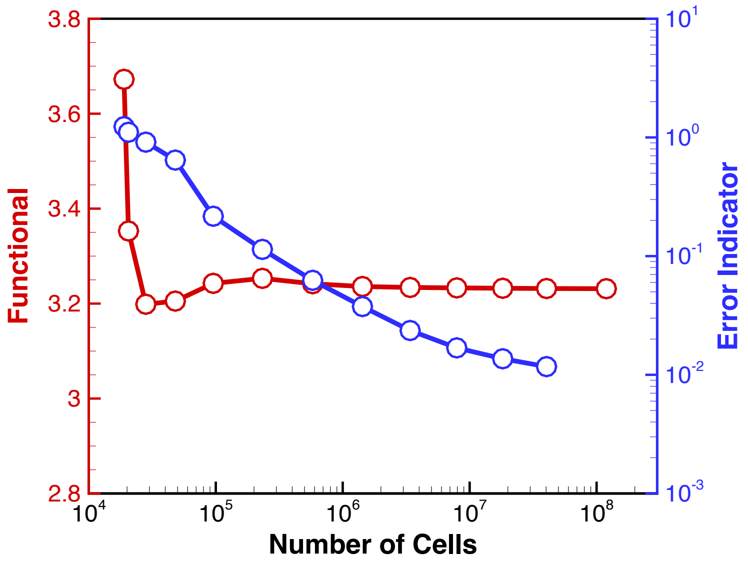

Figure 31. Mesh convergence of functional (lift + thrust + plume sensor) and error estimate on turbojet.

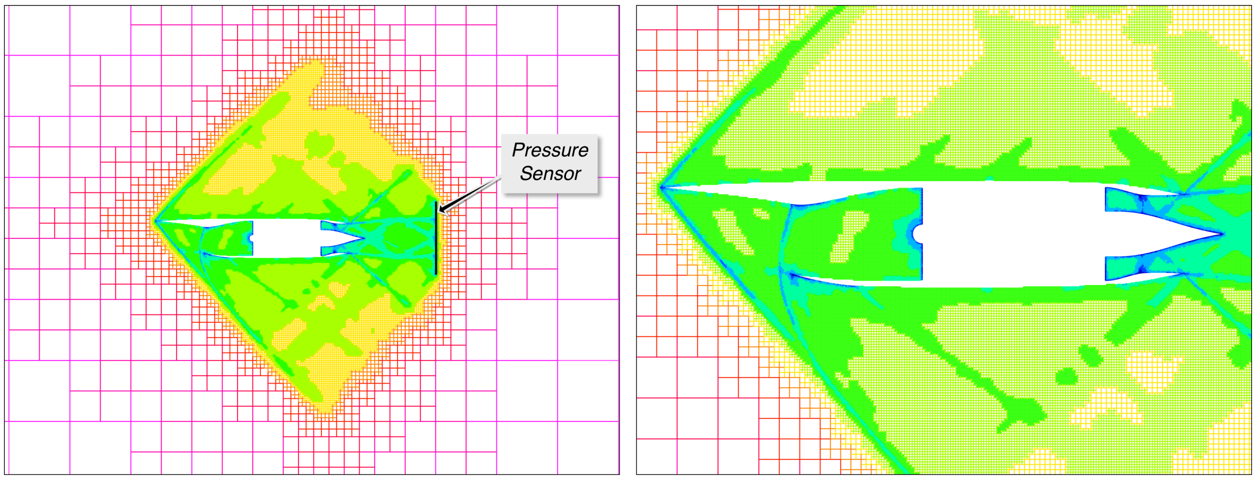

Figure 32. Adjoint-driven refined mesh (120 million cells) for turbojet with steered outflow rate to match the inflow rate. Colors represent cells of similar refinement.

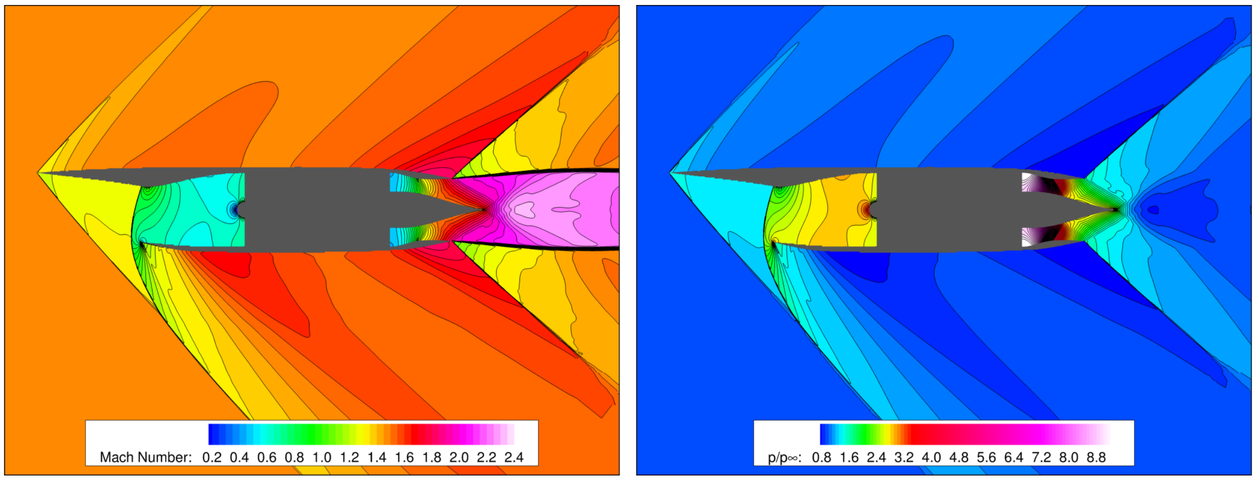

Figure 33. Mach number and pressure contours of flow through turbojet demonstrating steered outflow rate to match the inflow rate. M_∞ = 1.5, 120 million cells.

D. Scramjet

The three cases discussed above all required boundary conditions for subsonic inflow or outflow. However, there are instances when modeling a propulsion system that a supersonic inflow or outflow boundary condition is required. For instance, consider the notional scramjet shown in Fig. 34. For this geometry, the burner is not modeled and instead the boundary conditions are used to model the increase in total pressure and temperature of the core flow. The inflow/outflow boundary surfaces are shaded in cyan in Fig. 34.

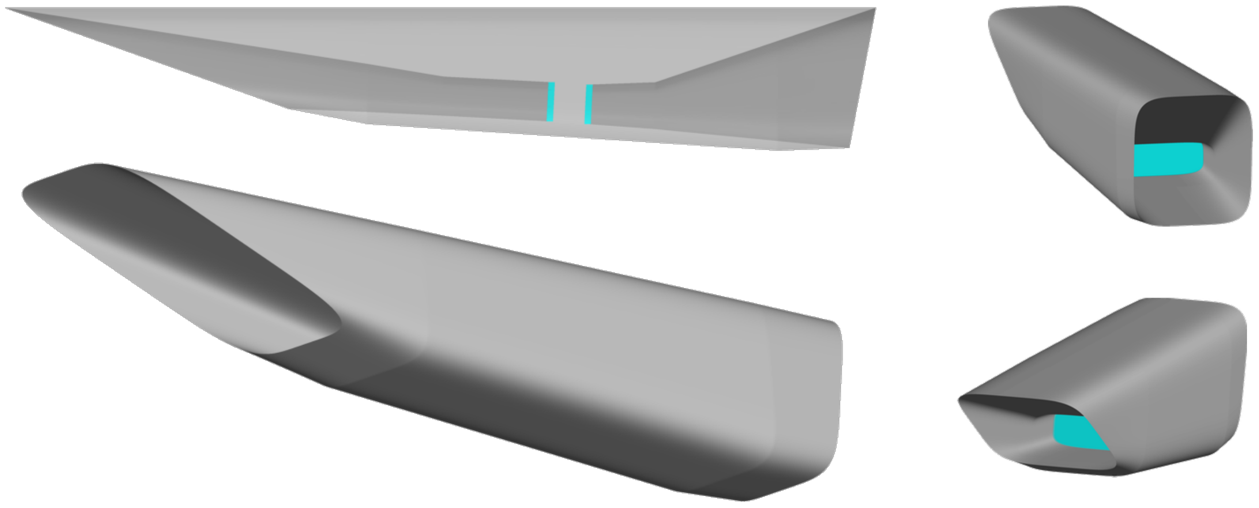

Figure 34. Notional scramjet geometry. Cyan faces represent where propulsion inflow/outflow boundary conditions are applied.

By definition, the flow through the scramjet core operating in hypersonic freestream conditions is never subsonic, and the propulsive inflow/outflow boundary conditions used in this model must realize this fact. In the inlet, the boundary condition is quite simple: extrapolation of all flow state values from the interior flow field. The back pressure subsonic outflow boundary condition would actually work here if the back pressure value were low enough to ensure supersonic flow. It works because the implementation does in fact use extrapolation of the entire interior flow state when exploiting the implemented safeguard. Likewise, the constant velocity outflow boundary condition would also work as long as the specified velocity results in a supersonic normal velocity. On the other hand, for the supersonic inflow boundary, none of the new boundary conditions presented in this work would suffice since they are strictly for subsonic inflow.

Of course, the power boundary conditions originally developed for Cart3D by Pandya et al.[2] are exactly what is needed to model the scramjet nozzle flow. As a demonstration, this original boundary condition was used on the scramjet model in Fig. 34 in Mach 5.0 freestream flow. The angle of attack is 2°. The boundary conditions were set to produce supersonic flow through both of the propulsion flow boundary faces. The boundary condition values were also set to match mass flow rates through both the propulsive inflow and outflow surfaces. This solution was generated using adjoint-driven mesh refinement, which represents a new feature beyond what was implemented by Pandya. For this case, the mesh was refined using a functional that includes a weighted sum of lift and thrust. A pressure line sensor is also used behind the scramjet to encourage the plume to be well-resolved by the mesh as was done with the supersonic turbojet in Section C. The output from this sensor is also included in the functional. The convergence history of this functional and the error estimate are given in Fig. 35. The resulting mesh is given in Fig. 36 and consists of over 220 million cells through 19 levels of refinement. The finest cell length is just over 0.0005% of the scramjet length. The Mach number and pressure contours on this solution are shown in Fig. 37. Note the highly refined mesh wherever a shock is present. The line sensor has also caused the mesh to be well refined in the plume region ahead of the pressure sensor.

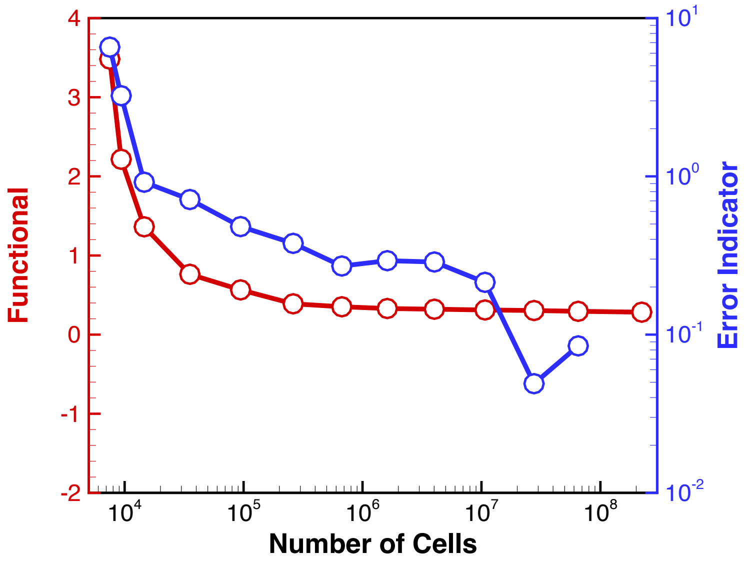

Figure 35. Mesh convergence of functional (lift + drag + plume sensor) and error estimate on scramjet.

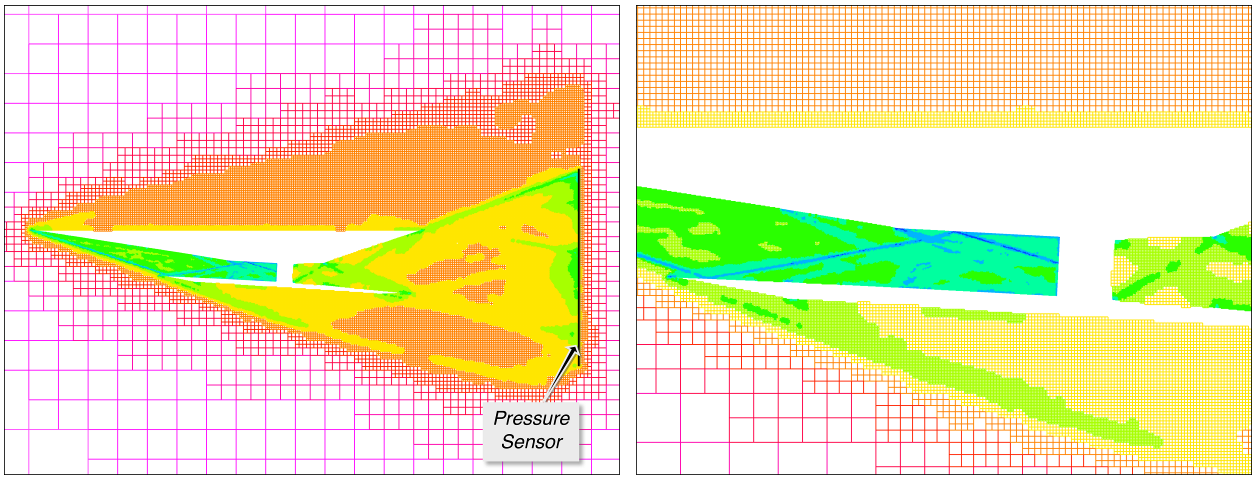

Figure 36. Adjoint-driven refined mesh (220 million cells) for scramjet. Colors represent cells of similar refinement.

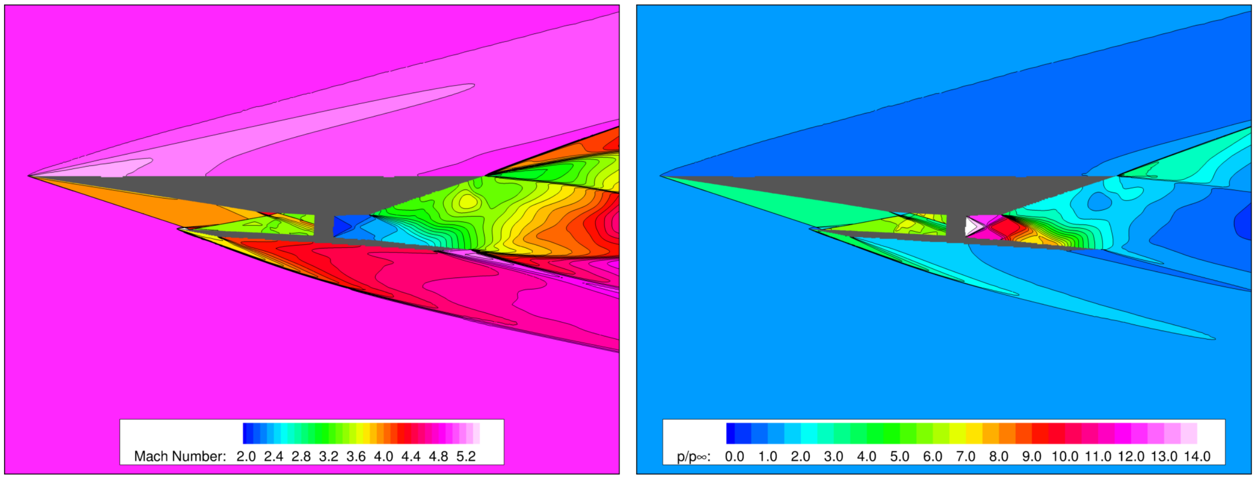

Figure 37. Mach number and pressure contours of flow through scramjet demonstrating the original power boundary condition. M_∞ = 5.0, 220 million cells.

E. Low Boom Supersonic Aircraft

As a final example case for the newly implemented boundary conditions, a supersonic aircraft was analyzed at a typical cruise condition (Mach 1.4 and 2.15° angle of attack). This design is a candidate to be a low boom demonstrator aircraft. The computational model of this aircraft has several boundary surfaces that require inflow/outflow boundary conditions that represent either a main propulsion flow stream or bypass flow used for various systems on the aircraft. These surfaces include a main inlet, an underwing inlet for cooling air, the main exhaust nozzle, an exhaust port under the nozzle itself, and a vent at the base of the vertical tail. These surfaces are shaded in cyan and explicitly labeled in Fig. 38. This example represents the application of the newly implemented inflow/outflow boundary conditions on a realistically complex full aircraft with multiple inlets and outlets.

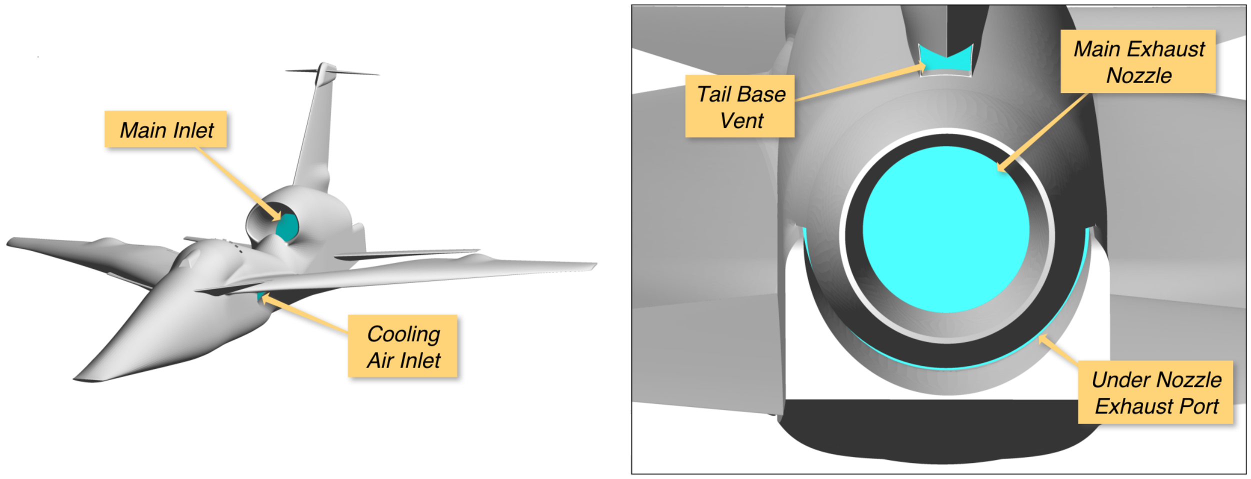

Figure 38. Low boom supersonic aircraft geometry. Cyan faces are inlets and outlets.

To analyze this aircraft, the back pressure outflow boundary condition was used in the inlets and the stagnation property inflow was used for the surfaces in the nozzle region. Adaptive mesh refinement was used for this solution with a functional of strictly drag (negative of thrust) of the aircraft. The convergence of the functional along with the error estimate is shown in Fig. 39. Convergence of the functional is excellent and the error estimate steadily declines as the mesh is refined. The final half body mesh is shown in Fig. 40 and consists of just over 70 million cells with 18 levels of refinement. The smallest cell length is about 0.0025% the length of the aircraft. The mesh has been highly refined in the regions of shocks. Since the freestream flow is supersonic and no pressure sensor was used behind the aircraft, the plume region mesh is not highly refined. Mach number contours of the solution are given in Fig. 41. Note the flow stream leaving the computational domain in the inlet in the left panel of Fig. 41. The right panel shows the flow entering the domain inside the nozzle, the vent below the nozzle, and the vent at the vertical tail base. The flow around this fully realized aircraft geometry is quite complex and the Cartesian method with adaptive mesh refinement was able to capture all the important flow features.

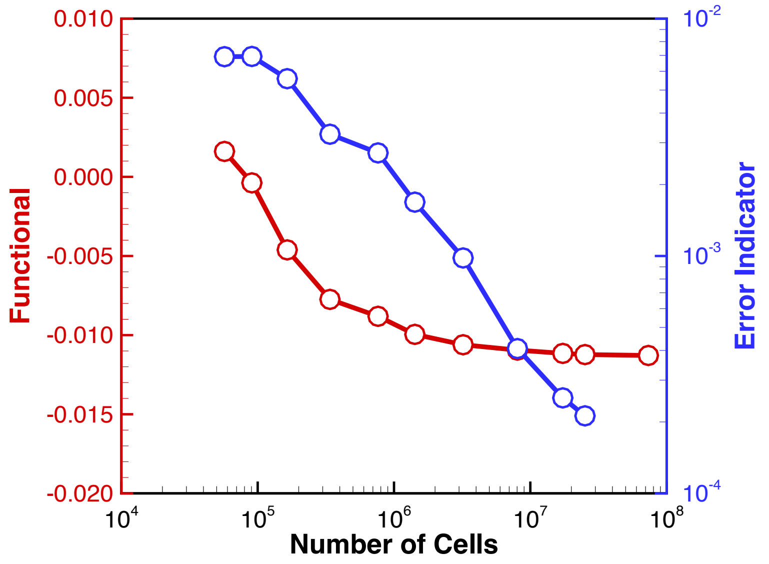

Figure 39. Mesh convergence of functional and error estimate on low boom aircraft.

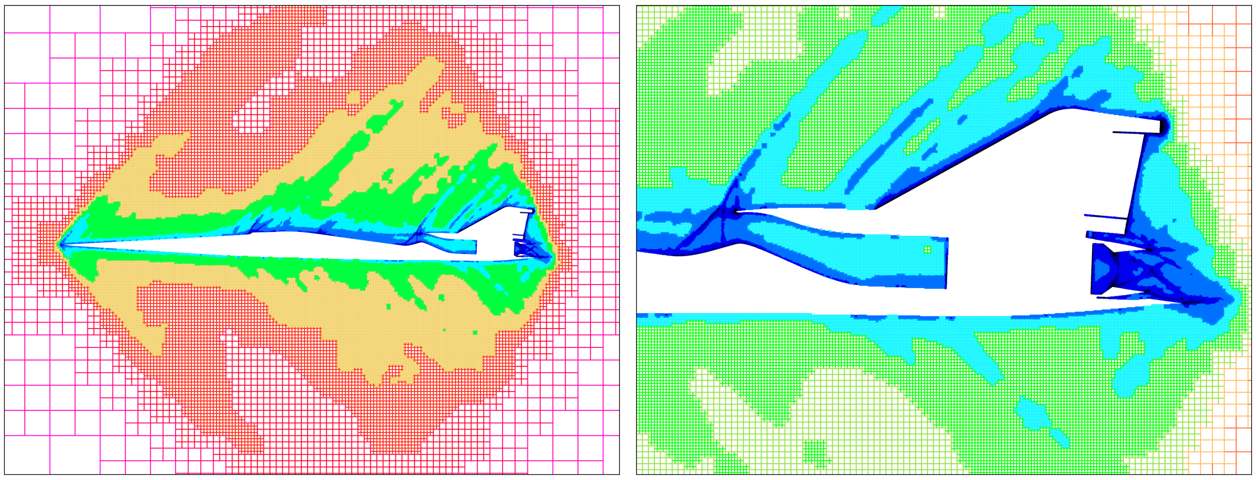

Figure 40. Adjoint-driven refined half-body mesh (70 million cells) for low boom aircraft. Colors represent cells of similar refinement.

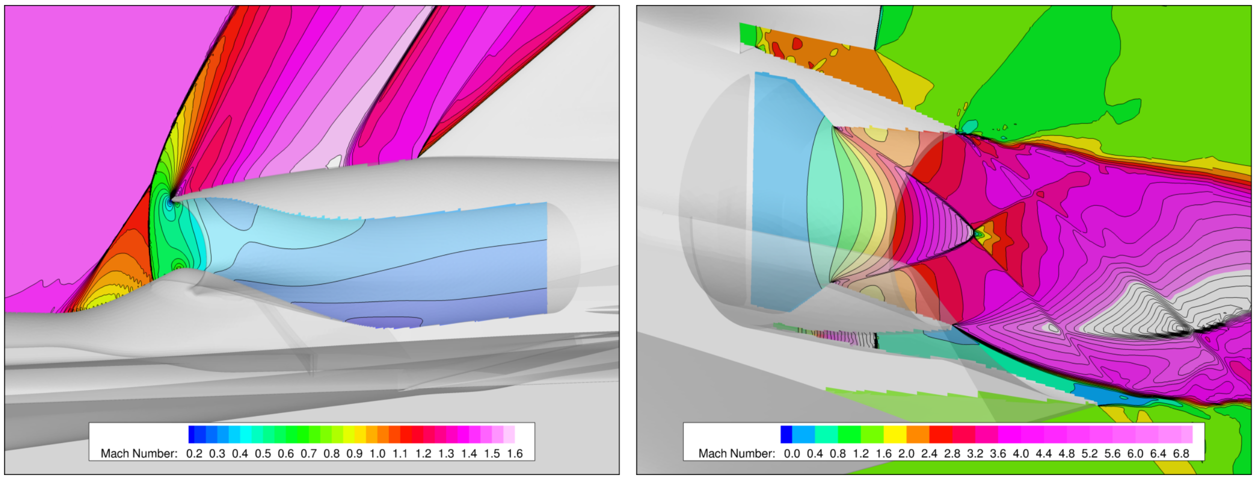

Figure 41. Mach number contours of flow surrounding low boom aircraft main inlet (left) and nozzle (right). M_∞ = 1.4, 70 million cells.

IV. Future Work

The new boundary conditions and their corresponding implementation into the adjoint solver represent a significant upgrade to the analysis capabilities and usability of the Cart3D flow solver. Currently, only pressure-based functionals are available to drive the mesh refinement, which may not be sufficient for accurate propulsion system analysis. Cart3D also has a design optimization capability.[21] Naturally, a next step is to extend the design capabilities of the Cart3D design framework to exploit the boundary conditions presented in this work. Such an extension would include developing new functionals that involve propulsion system design. When used for mesh refinement, these new functionals should also improve the accuracy of propulsion system simulations. Planned work will address these enhancements.

V. Conclusions

Four additional boundary conditions have been implemented into a Cartesian-based Euler analysis package. Two of these boundary conditions model subsonic flow exiting the computational domain through a specified surface while the other two model flow entering the domain. One outflow boundary condition permits the user to specify a back pressure while the other allows the user to specify an exiting normal flow velocity. Using the interior flow state next to the boundary, Riemann invariants are employed to determine the boundary state and hence the flux density tensor at the boundary. One of the inflow boundary conditions permits the user to specify both the total pressure and total temperature of the flow entering the domain. The other inflow boundary condition allows total temperature and a mass flow rate to be specified. Both of these inflow conditions assume the flow velocity vector entering the computational domain is normal to the boundary surface, thus providing the remaining required flow attributes for a consistent subsonic inflow boundary condition. All of these boundary conditions were implemented into the companion adjoint solver as well. Application of these new boundary conditions was demonstrated on three notional propulsion systems operating in low subsonic, transonic, and supersonic freestream flows. The previously implemented Riemann-solver-based boundary condition was also demonstrated on a notional scramjet operating in hypersonic flow to demonstrate how supersonic inflow and outflow continues to be handled by the flow solver. Finally, these boundary conditions were demonstrated on a modern low-boom supersonic transport design. All of these solutions were generated using adaptive mesh refinement that is driven by an adjoint solver in which the newly formulated boundary conditions are also properly implemented.

The constant velocity outflow boundary condition has also been coupled with a steering algorithm to allow the user to control the mass flow rate through an outflow boundary surface. This capability was demonstrated on the first three notional propulsion systems. The algorithm proved to be robust and fast. Mass-flow-rate control through an inflow boundary condition can be explicitly controlled by one of the newly implemented boundary conditions.

Acknowledgements

This research was supported by the NASA ARMD Commercial Supersonic Technology (CST) Project and by NASA Ames Research Center contract NNA16BD60C. Computing resources were provided by the NASA High-End Computing Program through the NASA Advanced Supercomputing Division at Ames Research Center.

References

[1]Aftosmis, M. J., Berger, M. J., and Adomavicius, G., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids With Embedded Boundaries," AIAA Paper 2000-0808, Jan. 2000. doi:10.2514/6.2000-808

[2]Pandya, S., Murman, S., and Aftosmis, M. J., "Validation of Inlet and Exhaust Boundary Conditions for a Cartesian Method," AIAA Paper 2004-4837, Aug. 2004. doi:10.2514/6.2004-4837

[3]Nemec, M. and Aftosmis, M. J., "Adjoint Sensitivity Computations for an Embedded-Boundary Cartesian Mesh Method," *Journal of Computational Physics*, Vol. 227, No. 4 (2008), pp. 2724-2742. doi:10.1016/j.jcp.2007.11.018

[4]Aftosmis, M. J., Berger, M. J., and Melton, J. E., "Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry," *AIAA Journal*, Vol. 36, No. 6 (1998), pp. 952-960. doi:10.2514/2.464

[5]van Leer, B., "Flux-Vector Splitting for the Euler Equations", *Lecture Notes in Physics*, Vol. 170 (Springer-Verlag, New York / Berlin, 1982), pp. 507-512. doi:10.1007/3-540-11948-5_66

[6]Aftosmis, M., Berger, M., and Murman, S., "Applications of Space-Filling-Curves to Cartesian Methods for CFD," AIAA Paper 2004-1232, Jan. 2004. doi:10.2514/6.2004-1232

[7]Nemec, M., Aftosmis, M., Murman, S., and Pulliam, T., "Adjoint Formulation for an Embedded-Boundary Cartesian Method," AIAA Paper 2005-877, Jan. 2005. doi:10.2514/6.2005-877

[8]Nemec, M. and Aftosmis, M., "Adjoint Error Estimation and Adaptive Refinement for Embedded-Boundary Cartesian Meshes," AIAA Paper 2007-4187, June 2007. doi:10.2514/6.2007-4187

[9]Yee, H. C., "Numerical Approximation of Boundary Conditions With Applications to Inviscid Equations of Gas Dynamics," NASA TM-81265, 1981.

[10]Pulliam, T. H. and Zingg, D. W., *Fundamental Algorithms in Computational Fluid Dynamics*, Springer, 2014, pp. 124-125.

[11]Hirsch, C., *Numerical Computation of Internal and External Flows, Computational Methods for Inviscid and Viscous Flow, Volume 2*, Wiley, Chichester, England, 1990, pp. 344-384.

[12]Bartels, R. E., Rumsey, C. L., and Biedron, R. T., "CFL3D Version 6.4-General Usage and Aeroelastic Analysis," 2006.

[13]Carlson, J.-R., "Inflow/Outflow Boundary Conditions with Application to FUN3D," NASA TM–2011-217181, 2011.

[14]Nichols, R. H. and Buning, P. G., "User's Manual for OVERFLOW 2.1," University of Alabama and NASA Langley Research Center, 2008.

[15]Allmaras, S. R., "A Coupled Euler/Navier-Stokes Algorithm for 2-D Unsteady Transonic Shock/Boundary-Layer Interaction," Ph. D. Dissertation, Aeronautics and Astronautics Dept., Massachusetts Institute of Technology, Cambridge, MA, 1989.

[16]Pearson, H. and McKenzie, A. B., "Wakes in Axial Compressors," *Journal of the Royal Aeronautical Society*, Vol. 63, No. 583 (Jul 1959), pp. 415-416.

[17]Reid, C., "The Response of Axial Flow Compressors to Intake Flow Distortion," *ASME 1969 Gas Turbine Conference and Products Show*, 1969.

[18]Gustafsson, B. and Oliger, J., "Stable Boundary Approximations for a Class of Time Discretizations of $u_t=AD_ou$," *Upsala University, Dept. of Computer Sciences, Report 87*, 1990.

[19]Nemec, M. and Aftosmis, M., "Toward Automatic Verification of Goal-Oriented Flow Simulations," NASA Rept. NASA/TM-2014-218386, 2014.

[20]Aftosmis, M., Nemec, M., and Cliff, S., "Adjoint-Based Low-Boom Design With Cart3D," AIAA Paper 2011-3500, June 2011. doi:10.2514/6.2011-3500

[21]Nemec, M. and Aftosmis, M., "Parallel Adjoint Framework for Aerodynamic Shape Optimization of Component-Based Geometry," AIAA Paper 2011-1249, Jan. 2011. doi:10.2514/6.2011-1249