Development of a Trajectory-Centric CFD-RBD Framework for Advanced Multidisciplinary/Multiphysics Simulation

Zachary J. Ernst*, Madilyn K Drosendahl †, Bradford E. Robertson‡,
Dimitri N. Mavris§
Aerospace Systems Design Lab, Georgia Institute of Technology, Atlanta, GA 30332

The desire to model increasingly complex unsteady flow phenomena drives coupling of physics-based disciplinary analysis tools, such as coupled aerodynamics-rigid body dynamics simulations. This paper documents the creation of a framework linking the six-degree-of-freedom trajectory propagator POST2 with NASA's FUN3D computational fluid dynamics flow solver. Cross-code verification between the framework and a CFD-centric 6DOF code is performed using the Army-Navy Finner projectile experiencing unsteady accelerating flow. Free-flight simulations of an entry vehicle ballistic range test are validated against physical and computational experiments.

Nomenclature

a	speed of sound	Δt_{fs}	flow solver physical time step
C_A, C_Y, C_N	axial, side, and normal force coefficients	TR	jet total temperature ratio
C_{LL}, C_M, C_{LN}	roll, pitch, and yaw moment coefficients	$\bar{v}$	velocity vector
d	diameter	$\bar{X}$	vehicle state vector
$\bar{F}$	force vector		
$\boldsymbol{I}$	inertia tensor	γ	flight path angle
I_{xx}, I_{yy}, I_{zz}	roll, pitch, and yaw inertia components	λ, Φ	longitude and geodetic latitude
L	length in dimensional units	ϕ, θ, ψ	relative roll, pitch, and yaw Euler angle
L_{grid}	reference length in grid units	ϕ_I, θ_I, ψ_I	inertial roll, pitch, and yaw Euler angle
$\bar{M}$	moment vector	Ψ	azimuth angle
M	Mach number	$\bar{\omega}$	angular velocity vector
m	mass	ω_{earth}	planetary rotation rate
P	static pressure	$\bar{\omega}_{rel}$	vertical-relative angular velocity vector
P_t	total pressure		
PR	jet total pressure ratio	*Subscripts:*	
Q	dynamic pressure	B	Body frame
$\bar{q}$	quaternion vector	cmd	commanded
$\boldsymbol{R}_{ij}$	j to i frame rotation matrix	E	Earth-centered rotating frame
$\bar{r}$	position vector	F	Body reference frame
$\Delta\bar{r}$	vehicle C.G. translation vector	I	Inertial frame
$\Delta\bar{r}_g$	grid translation vector	O	Observer frame
s	freestream speed	P	Intermediary frame
T	static temperature	$plenum$	at the nozzle plenum
T_t	total temperature	ref	reference parameter
t	time	V	Local vertical frame
Δt	trajectory propagator physical time step		

*Senior Graduate Researcher, ASDL, School of Aerospace Engineering, Georgia Tech, AIAA Student Member
†Graduate Researcher, ASDL, School of Aerospace Engineering, Georgia Tech, AIAA Student Member
‡Research Engineer II, ASDL, School of Aerospace Engineering, Georgia Tech, AIAA Member
§S.P. Langley Distinguished Regents Professor and Director of ASDL, Georgia Tech, AIAA Fellow

I. Introduction

Flight simulation is a critical design activity for aerospace vehicles, used for trajectory optimization [1], dispersion analysis [2], guidance development [3], and many other tasks. A flight simulation models the relevant disciplines — such as flight dynamics, aerodynamics, propulsion, atmospherics, controls, etc. — as they act on the vehicle over time. The trajectory propagator acts as the focal point of the simulation: calling disciplinary models, integrating the equations of motion, and updating the vehicle state. These models are traditionally provided as independent, self-contained modules which are surrogates for one or more higher-fidelity models[4]. For example, the aerodynamics model may provide the force and moment coefficients as a database populated from physical [5] or computational experiments [6]. These models achieve a low computational cost by a reduction in fidelity, necessary due to the large number of flight simulations that must be performed. Each model may be constructed by different organizations or at different stages of the design process [4].

While the accuracy of these simulations is sufficient for most cases, designers may require higher-fidelity simulation during particularly complex regimes of flight such as separation events [7] or accelerating through the transonic regime [8]. These models might also be limited in their capability to model coupling between disciplines, such as fluid-structure interactions (FSI) [9] or aerodynamic/control interaction [10]. This has led to the goal of developing advanced multidisciplinary/multiphysics simulation capabilities [11].

A coupled multiphysics model directly links disciplinary codes to the trajectory simulation, running at the exact conditions requested by the trajectory code. In addition to capturing the aforementioned interactions, a coupled model eliminates both the loss of fidelity due to the use of a surrogate model and the interpolation error inherent with databases. It can also decrease *a priori* computing cost if configuration or geometry is updated, or if off-nominal (e.g., abort) scenarios must be analyzed — scenarios which would otherwise require new or updated databases.

One of the most commonly used coupled models links the aerodynamics and flight dynamics for free-flight simulation. Dynamic, time-accurate computational fluid dynamics (CFD) models are coupled with a rigid body dynamics (RBD) simulation to move the vehicle in full six-degree-of-freedom (6DOF) motion. The continuous integration of the flow field preserves flow history, which means that unsteady aerodynamic effects are modeled. CFD-RBD simulations are used regularly in the field of subsonic and supersonic projectile design. Store separation relies heavily on CFD-RBD simulation to determine whether a projectile or other object released by an aircraft might be at risk of recontact [12]. Meakin developed some of the earliest simulations for time-accurate Navier-Stokes coupled with 6DOF for modeling store separation [13]. Coupled CFD-RBD simulations known as "virtual fly-out" have also been used to estimate aerodynamic coefficients for projectiles [14–16]. Sahu has performed coupled CFD-RBD simulations for a variety of finned projectiles [17], adding guidance and control system models for open-loop [18] and closed-loop [19] controller simulation.

Murman et al. developed CFD-RBD simulations in the OVERFLOW solver for the simulation of atmospheric entry vehicles [20]. Stern et al. developed "free-flight CFD" within the US3D flow solver to predict static and dynamic coefficients for the Mars Science Laboratory (MSL) entry vehicle [21]. Stern et al. also used the CFD-RBD simulations to predict surface pressure at specific points for comparison to flight instrumentation [22]. More recently, Brock et al. used the same framework to recreate the ballistic range tests for the Supersonic Dynamics Flight Test (SDFT) with good agreement to the experimental results [23].

These existing CFD-RBD models have generally been constructed as an expansion of the flow solver. The RBD simulations are often limited in complexity to a particular 6DOF numerical integration scheme [20] or gravitational model [24]. Constructing a trajectory propagator-centric multidisciplinary/multiphysics simulation would enable the use of state-of-the-art flight simulation tools [25]. This approach makes it easier to link CFD-RBD models with other disciplinary models that are already configured to work with a trajectory propagator. The simulations could also utilize the analysis features of the propagator, such as trajectory optimization.

This paper details the development of a multiphysics simulation framework that couples Program to Optimize Simulated Trajectories II (POST2), a state-of-the-art trajectory propagation code, with the FUN3D CFD flow solver. The framework is designed to enable the simulation of multiple problems of interest in the supersonic and low hypersonic regimes, with the most complex being an atmospheric entry vehicle with active reaction control system (RCS) jets. Cross-code verification is performed against an existing CFD-RBD code built in FUN3D, through simulations of a projectile in a ballistic drop, as a cross-check of the data transformation. Validation is performed against ballistic range tests for an entry vehicle with nontrivial initial conditions.

II. CFD/RBD Model

The CFD-RBD model solves the fluid flow equations simultaneously with the flight dynamics equations of motion. The equations of motion are integrated forward in time to calculate the vehicle state, and the external aerodynamic forces and moments are calculated using CFD to solve the flow equations.

The vehicle is modeled as a rigid body under the effect of external forces and moments. These include the effect of aerodynamics, gravitation, propulsion and control. The vehicle can move in six degrees of freedom (three in translation and three in rotation) with respect to an inertial reference frame. The inertial frame I and body frame B are defined as shown in Figure 1. The body frame is fixed to the vehicle with its origin at the center of mass, the x-axis pointing towards the front of the vehicle, and the y- and z-axes forming a right-handed reference frame. The vehicle state $\bar{X}$ consists of: $\bar{r}_I$, its position vector of the center of gravity in the inertial frame; $\bar{v}_B$ and $\bar{\omega}_B$, its inertial linear and angular velocity vectors, respectively, represented in the body frame; and ϕ_I, θ_I, and ψ_I, the roll, pitch, and yaw Euler angles, respectively. For simplicity, the equations below are presented for a vehicle with constant mass properties.

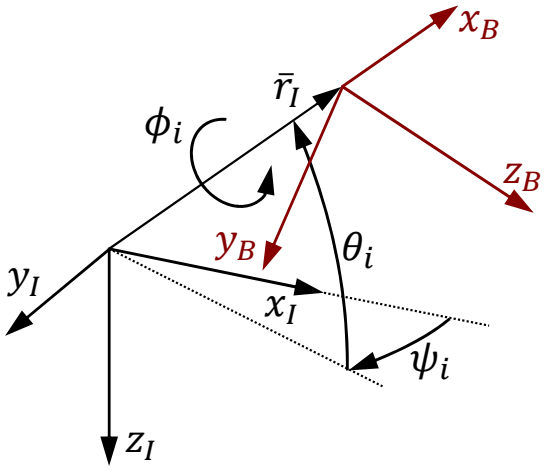


Fig. 1 Inertial and Body Reference Frames

The equations of motion govern changes to the vehicle state over time [26]:

$$\dot{\bar{v}}_B = \frac{1}{m}(\bar{F}_B - \dot{m}\bar{v}_B) - \bar{\omega}_B \times \bar{v}_B \tag{1}$$

$$\dot{\bar{r}}_I = \boldsymbol{R}_{IB}\bar{v}_B \tag{2}$$

$$\dot{\bar{\omega}}_B = \boldsymbol{I}_B^{-1}(\bar{M}_B - \bar{\omega}_B \times (I_B\bar{\omega}_B) - \dot{\boldsymbol{I}}_B\bar{\omega}_B) \tag{3}$$

$$\begin{bmatrix}\dot{\phi}_i\\ \dot{\theta}_i\\ \dot{\psi}_i\end{bmatrix} = \begin{bmatrix}1 & \sin\phi_I\tan\theta_I & \cos\phi_I\tan\theta_I\\ 0 & \cos\phi_I & -\sin\phi_I\\ 0 & \sin\phi_I\sec\theta_I & \cos\phi_I\sec\theta_I\end{bmatrix}\bar{\omega}_B \tag{4}$$

$\bar{F}_B$ and $\bar{M}_B$ are the external force and moment vectors, respectively. m is the vehicle mass, and $\boldsymbol{I}_B$ is the inertia tensor. A dot over the variable represents its derivative with respect to time. $\boldsymbol{R}_{IB}$ is the rotation matrix defining the transformation between the body and inertial frames:

$$\boldsymbol{R}_{IB} = \begin{bmatrix}\cos\psi_I\cos\theta_I & \cos\psi_I\sin\phi_I\sin\theta_I - \cos\phi_I\sin\psi_I & \sin\phi_I\sin\psi_I + \cos\phi_I\cos\psi_I\sin\theta_I\\ \cos\theta_I\sin\psi_I & \cos\phi_I\cos\psi_I + \sin\phi_I\sin\psi_I\sin\theta_I & \cos\phi_I\sin\psi_I\sin\theta_I - \cos\psi_I\sin\phi_I\\ -\sin\theta_I & \cos\theta_I\sin\phi_I & \cos\phi_I\cos\theta_I\end{bmatrix} \tag{5}$$

Together, these form a set of 12 ordinary differential equations. Numerical integration is used to solve the equations of motion and propagate the vehicle state forward in time according to the initial value problem in Equation 6.

$$\dot{\bar{X}} = f(\bar{X}, t), \quad \bar{X}(t_0) = \bar{X}_0 \tag{6}$$

The Runge-Kutta 4th-order integration scheme is used to calculate the update to the vehicle state based on a series of substeps [27]:

$$\bar{X}(t+\Delta t) = \bar{X}(t) + \frac{\Delta t}{6}(\bar{k}_1 + 2\bar{k}_2 + 2\bar{k}_3 + \bar{k}_4) \tag{7}$$

where Δt is the physical time step used by the trajectory propagator, and $\bar{k}_1$ to $\bar{k}_4$ are the derivatives at each of the substeps:

$$\begin{aligned} \bar{k}_1 &= f(\bar{X}, t) \\ \bar{k}_2 &= f(\bar{X} + \Delta t\frac{\bar{k}_1}{2}, t + \frac{\Delta t}{2}) \\ \bar{k}_3 &= f(\bar{X} + \Delta t\frac{\bar{k}_2}{2}, t + \frac{\Delta t}{2}) \\ \bar{k}_1 &= f(\bar{X} + \Delta t\bar{k}_3, t + \Delta t) \end{aligned} \tag{8}$$

The updated vehicle state is used to update the grid motion in the flow solver.

The fluid flow is governed by the Navier-Stokes equations, written using the arbitrary Lagrangian-Eulerian (ALE) formulation in Equation 9 [28]:

$$\frac{\partial}{\partial t}\int_{\mathcal{V}} \boldsymbol{q}\, d\mathcal{V} + \oint_{\partial\mathcal{V}} (\boldsymbol{F}^* - \boldsymbol{F}_{\mathcal{V}}) \cdot \hat{n} d\mathcal{S} = 0 \tag{9}$$

where $\boldsymbol{q}$ represents the conserved variables and $\mathcal{V}$ is the control volume, bounded by control surface $\partial\mathcal{V}$ with local control volume face velocity $\bar{W}$. $\boldsymbol{F}^*$ and $\boldsymbol{F}_{\mathcal{V}}$ are the convective and diffusive fluxes of $\boldsymbol{q}$, respectively. The ALE formulation allows for the prediction of unsteady aerodynamics as the vehicle moves in time. The flux through a moving control volume must account for augmented or reduced flux through the control surface due to the local control surface speed:

$$\boldsymbol{F}^* = \boldsymbol{F} - \boldsymbol{q}\bar{W}^T \tag{10}$$

A CFD flow solver is used to solve these equations for the time-accurate flow in a volume around the vehicle. The volume is spatially discretized into a grid, with the convective fluxes computed using a flux-splitting scheme [29]. The solution is advanced in time using a dual time-stepping approach [30]. The outer time step Δt_{fs} represents a physical discretization in time that is applied to the entire grid. This time step is equal to the time step used in the numerical integration of the equations of motion.The inner time step is used to help solve the physical flow equations. This step is varied spatially and is subject to relaxation to reduce the residuals of $\boldsymbol{q}$. Once the solution is converged, the pressure and shear forces acting on the surface of the vehicle are integrated into force and moment coefficients. These coefficients are used to update the external forces and moments in the equations of motion.

This method of coupling can be characterized as a modified, staggered nonlinear block-Gauss-Seidel algorithm [31]. The trajectory propagator steps forward with one iteration. Then, the flow solver executes some number of time steps and subiterations until it is converged. The time step Δt is small enough that the solution of the equations of flow and motion are assumed to converge at each step. Consider representative values for Δt and Δt_{fs} nondimensionalized by s/L_{ref}, the time taken by a fluid particle at the freestream velocity to travel over a reference length. Whereas the nondimensional step used in CFD-RBD simulations may be on the order of 10^{-2} [23], the nondimensional step used in trajectory simulations without CFD is usually on the order of 10^0 [32]. By running at the smaller time step required by the CFD, a sufficient degree of convergence can be achieved.

In addition, the aerodynamic force and moment coefficients are held constant during integration across the time-integration substeps. Using constant aerodynamic forces and moments allows the flow solver to keep a single flow field in memory and preserve the monotonic nature of the solution. While using constant force and moment coefficients reduces the order of accuracy of the integration scheme, the time step is orders of magnitude smaller than typical RBD time steps, mitigating the error propagation [27]. This is a common approach for CFD-RBD simulation [20, 21, 24].

III. Framework Construction

The framework consists of the trajectory propagator, the CFD flow solver, and the interface which links the two codes. Independent processes are used to run the flow solver and trajectory propagator.

A. Flow Solver

FUN3D is a production flow analysis and design tool developed at the NASA Langley Research Center [33]. From its inception in the late 1980s, FUN3D has been developed into a suite of tools for flow analysis, mesh adaptation, and design optimization [29, 34]. The flow solver is capable of solving the Reynolds-Averaged Navier-Stokes (RANS) or Euler equations for steady-state or time-accurate flow. FUN3D uses a node-based finite-volume discretization capable of solving unstructured or mixed-element meshes. It can model perfect gas or a generic gas with multiple chemical species; in compressible or incompressible flows; at thermochemical equilibrium or non-equilibrium. FUN3D uses a second-order-accurate spatial discretization, with options for first- to fourth-order temporal discretization (with temporal error controllers). The software includes a variety of turbulence models, including modification of RANS for Detached Eddy Simulation (DES), and flux splitting schemes.

FUN3D was selected as the flow solver based on its capability to model problems of interest in the hypersonic and supersonic flight regime. FUN3D has a long history of usage in government and industry projects, including for analysis of the Phoenix and MSL entry vehicles [10, 35]. FUN3D can model boundary conditions for internal flow and propulsion simulation, including inlets and nozzles. Critically, FUN3D uses the ALE formulation of the governing equations, allowing for freeform mesh movement and deformation [28]. Rigid mesh movement allows for free-form translation and rotation of all points in the mesh in unison, whereas mesh deformation changes the relative position and orientation between nodes. Rigid motion and deformation can be combined in the same simulation. FUN3D also includes a Python-based application programming interface (API), which allows external control of the execution down to the iteration level.

B. Trajectory Propagator

Although FUN3D can be compiled with a 6DOF RBD trajectory propagator [24], this RBD code has limited capabilities in the selection of gravitational models, numerical integration schemes, or initial conditions. It also does not allow for coupling with other disciplines. An alternative implementation uses an external process to perform the RBD simulation and drive the motion in the CFD flow solver. This external motion driver allows the flow solver to be linked with a state-of-the-art trajectory propagation software, with greatly expanded capability — it can be coupled with other disciplines such as propulsion or guidance, navigation, and control (GN&C).

POST2 is a trajectory propagation and optimization program also developed at the Langley Research Center [36]. POST2 models the trajectory of one or more point masses in flight about a single attracting body, inside or outside of an atmosphere. The propagator can solve problems either in 3DOF by integrating the translational equations of motion or in 6DOF by integrating the rotational equations of motion as well. The planetary model is generalized, with arbitrary rotational, gravitational (mass and oblateness), and atmospheric parameters. POST2 includes discrete parameter targeting and optimization for customizable trajectory parameters.

POST2 is the state of the art in trajectory propagation software, used for trajectory analysis of orbital launch vehicles [1], and Earth and Mars entry vehicles [2, 37, 38]. POST2 is designed with great flexibility in enabling custom code modules. Simulations are run with POST2 acting as the integrator for disciplinary models, such as aerodynamics, GN&C, and propulsion that are specifically built for the vehicle under study.

C. Communication

A flowchart of the framework is laid out in Figure 2, showing the communication between POST2 and FUN3D. Network sockets are used to handle communication between the C-based POST2 and a Python3-based wrapper. This wrapper performs data transformation and controls the execution of FUN3D through the API.

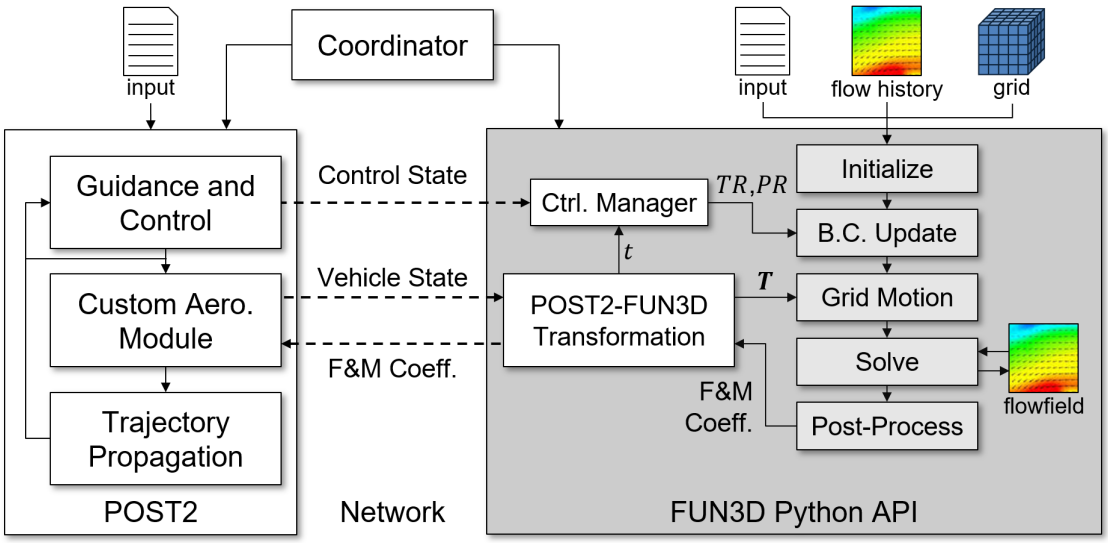


Fig. 2 POST2-FUN3D Framework Flowchart

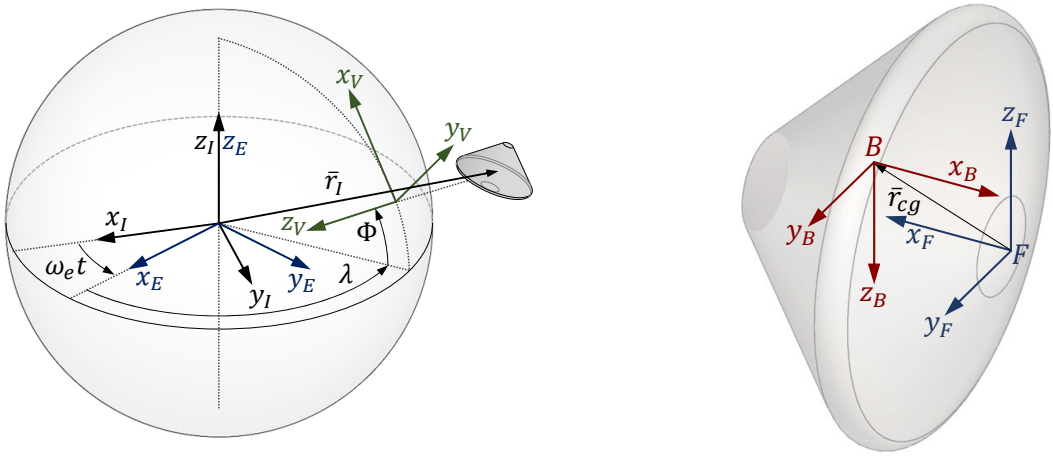
A coordinator manages the initialization and termination of the independent POST2 and FUN3D processes. When the aerodynamic coefficients are queried in POST2, a custom aerodynamic module is run that sends the current vehicle state to the FUN3D wrapper. The state is transformed into the reference frames required for FUN3D input and nondimensionalized. If the simulation includes active RCS, the control system state is similarly communicated to the wrapper. If the flow solver is running at a smaller time step than the trajectory propagator, the wrapper interpolates between the current and previous states. The control state is used to update the inlet boundary conditions, and the vehicle state is used to update the grid motion. The flow solver is then executed for one or more time-accurate steps (interpolating the states each time if necessary) until the physical time in POST2 and FUN3D match. The solution is post-processed to calculate the force and moment coefficients, which are transformed back into the POST2 reference frames and communicated to POST2.

D. Data Transformation

Data must be transformed between the POST2 and FUN3D reference frames, which are defined in Figures 3 and 4. For most CFD-RBD models, the inertial frame I is assumed to be the local horizontal frame in a "Flat Earth" model with a uniform gravitational field. However, for a spheroidal planetary model with nonzero rotation, I represents the Earth-centered inertial (ECI) frame. The ECI frame is oriented with the x-axis pointing through the equator and 0° longitude at the epoch, the y-axis pointing through the equator at a perpendicular direction, and the z-axis pointing North through the planet's axis of rotation. As the planet rotates with angular velocity ω_e it carries with it the Earth-centered rotating frame E, whose x- and y-axes remain pointing through 0° and 90° longitude, respectively.

The local vertical frame V has its origin on the planet's surface directly below the vehicle. Its x- and y-axes lie in the horizontal plane with the x-axis pointing North. The z-axis lies along the local vertical pointing inward towards the planet (note that for an oblate spheroid this may not point directly at the planet's center). The orientation of V with respect to E is given in terms of the longitude λ and geodetic latitude Φ angles.

The Body Reference frame F is used to define the vehicle with respect to the outer mold line. The origin of F is commonly placed at the nose of the vehicle, with the x-axis pointing aft, the y-axis pointing right, and the z-axis pointing up (equivalent to a 180° rotation about the y-axis relative to the B frame).



(a) ECI, ECR, and Local Vertical Frames **(b) Body and Body Reference Frames**

Fig. 3 POST2 and FUN3D Reference Frames

Translation and orientation are defined in FUN3D with respect to an observer frame O, which moves at a constant velocity relative to the atmosphere. However, the orientation of the O frame is arbitrary. It is therefore convenient to define an intermediary frame P as shown in Figure 4. The origin of this frame is coincident with the initial local vertical frame V_0, with the x-axis pointing along the vehicle's initial velocity vector v_{0,V_0} and the y-axis along the local horizontal. The orientation of P with respect to V_0 is defined by the azimuth angle Ψ and flight path angle γ. The O frame is defined as a 180° rotation about the y-axis from the P frame. Using the intermediary frame decouples specification of freestream conditions in FUN3D from the initial orientation within POST2, making it possible to change the azimuth and flight path of the vehicle without having to recreate the initial flow field.

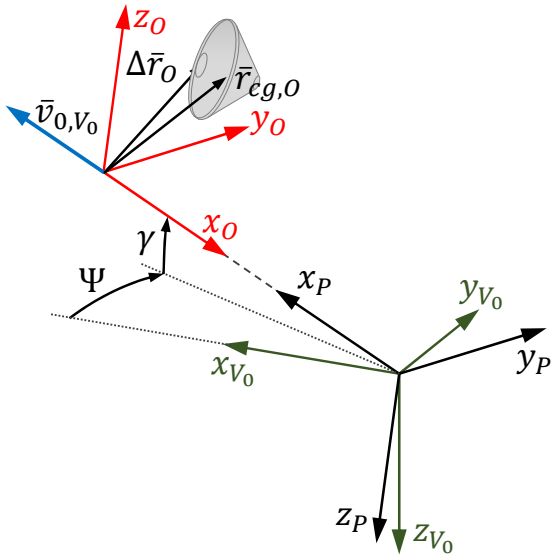


Fig. 4 Intermediate and Observer Frames

1. Angular Components

FUN3D requires the orientation of the vehicle to be input as a rotation matrix between the O and F frames. This rortation matrix from F to O is calculated using Equation 11:

$$\boldsymbol{R}_{OF} = \boldsymbol{R}_{OP}\boldsymbol{R}_{PB}\boldsymbol{R}_{BF}, \qquad \boldsymbol{R}_{OP} = \boldsymbol{R}_{BF} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{11}$$

where $\boldsymbol{R}_{PB}$ is the transformation between the Intermediary and Body frames. $\boldsymbol{R}_{PB}$ is calculated using Equation 12:

$$\boldsymbol{R}_{PB} = \boldsymbol{R}_{PV_0}\boldsymbol{R}_{V_0E}\boldsymbol{R}_{EV}\boldsymbol{R}_{VB} \tag{12}$$

$\boldsymbol{R}_{PV_0}$ is the rotation matrix between the intermediate and initial vertical frames:

$$\boldsymbol{R}_{PV_0} = \begin{bmatrix} \cos\Psi\cos\gamma & \cos\gamma\sin\Psi & -\sin\gamma \\ -\sin\Psi & \cos\Psi & 0 \\ \cos\Psi\sin\gamma & \sin\Psi\sin\gamma & \cos\gamma \end{bmatrix} \tag{13}$$

where γ and Ψ are available from POST2. $\boldsymbol{R}_{V_0E}$ and $\boldsymbol{R}_{EV}$ are the 3-2 rotation matrices calculated using the initial and current geodetic latitude and longitude, respectively.

$$\boldsymbol{R}_{V_0E} = \begin{bmatrix} -\cos\lambda_0\sin\Phi_0 & -\sin\Phi_0\sin\lambda_0 & \cos\Phi_0 \\ -\sin\lambda_0 & \cos\lambda_0 & 0 \\ -\cos\Phi_0\cos\lambda_0 & -\cos\Phi_0\sin\lambda_0 & -\sin\Phi_0 \end{bmatrix} \tag{14}$$

R_{VB} is the 3-2-1 rotation matrix calculated from the relative Euler angles ϕ, θ, and ψ, with the same form as Equation 5. The Euler angles, latitude, and longitude are available directly within POST2.

The angular velocity in the F frame is calculated using Equation 15:

$$\bar{\omega}_{rel,F} = \boldsymbol{R}_{FI}(\bar{\omega}_I - [0 \quad 0 \quad \omega_{earth}]^T) \tag{15}$$

where $\bar{\omega}_{rel,B}$ is the angular velocity of the vehicle relative to the V frame, available directly from POST2, and ω_{earth} is the magnitude of the planetary rotation rate. The rotation matrix $\boldsymbol{R}_{FI}$ is:

$$\boldsymbol{R}_{FI} = \boldsymbol{R}_{FB}\boldsymbol{R}_{BV}\boldsymbol{R}_{VE}\boldsymbol{R}_{EI} \tag{16}$$

2. Linear Components

FUN3D requires a center of gravity position and a grid translation vector, both in the O frame. POST2 provides the position vector in the I frame. The position in the ECR frame, $\bar{r}_E$, is calculated using Equation 17:

$$\bar{r}_E = \boldsymbol{R}_{EI}\bar{r}_I \tag{17}$$

where $\boldsymbol{R}_{EI}$ is the transformation between the ECI and Earth-centered, Earth-fixed (ECEF) frames. This rotation matrix is calculated as:

$$\boldsymbol{R}_{EI} = \begin{bmatrix} \cos\omega_{earth}t & \sin\omega_{earth}t & 0 \\ -\sin\omega_{earth}t & \cos\omega_{earth}t & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

where t is the time since epoch (recall that the ECEF and ECI frames are aligned at epoch in POST2). The displacement is calculated relative to the initial position:

$$\Delta\bar{r}_E = \bar{r}_E - \bar{r}_{0,E} \tag{19}$$

Transforming the displacement into the observer frame must account for that frame's velocity. The position in the V_0 frame is calculated using Equation 20:

$$\Delta\bar{r}_{V_0} = \boldsymbol{R}_{V_0E}\Delta\bar{r}_E - \bar{v}_{0,V_0}t \tag{20}$$

where $\bar{v}_{0,V_0}$ is calculated as:

$$\bar{v}_{0,V_0} = s_0[\cos\gamma\cos\Psi \quad \cos\gamma\sin\Psi \quad -\sin\gamma]^T \tag{21}$$

where s_0 is the initial freestream speed. The position in the O frame is calculated using Equation 22:

$$\bar{r}_O = \boldsymbol{R}_{OP}\boldsymbol{R}_{PV_0}\Delta\bar{r}_{V_0} + \bar{r}_{cg,0,O} \tag{22}$$

where $\bar{r}_{cg,0,O}$ is the initial position of the center of gravity in the O frame. The grid translation vector is calculated as:

$$\bar{r}_{g,O} = \Delta\bar{r}_O - \boldsymbol{R}_{OF}\bar{r}_{cg,F} \tag{23}$$

where $\bar{r}_{cg,F}$ is the center of gravity in the body reference frame.

The velocity vector is calculated using Equation 24:

$$\bar{v}_O = \boldsymbol{R}_{OE}(\boldsymbol{R}_{EI}\bar{v}_I - \bar{r}_E \times [0 \quad 0 \quad \omega_{earth}]^T), \qquad \boldsymbol{R}_{OE} = \boldsymbol{R}_{OP}\boldsymbol{R}_{PB}\boldsymbol{R}_{BE} \tag{24}$$

The position and translation are non-dimensionalized by the ratio L_{grid}/L_{ref} where L_{grid} and L_{ref} are the reference length of the vehicle in grid units and dimensional units, respectively. The velocity is non-dimensionalized by $1/a_{ref}$, where a_{ref} is the reference speed of sound in the freestream.

3. Jet Plenum Conditions

For each jet i, POST2 sends the commanded total chamber pressure and temperature $P_{i,t,cmd}$ and $T_{i,t,cmd}$. Research into time-accurate simulation of a jet in supersonic crossflow has shown that the stability of the flow solver is sensitive to large, instantaneous changes in plenum pressure [39]. Therefore, the plenum total pressure is ramped according to Equation 25 to maintain flow solver stability [39]:

$$P_{i,t,plenum} = \begin{cases} \min\left(P_{i,t,prev} + k_{p+}\Delta t_{fs}, \;\; P_{i,t,cmd}\right), & P_{i,t,cmd} \geq P_{i,t,prev} \\ \max\left(P_{i,t,prev} - k_{p-}\Delta t_{fs}, \;\; P_{i,t,cmd}\right), & P_{i,t,cmd} < P_{i,t,prev} \end{cases} \tag{25}$$

where k_{p+} and k_{p-} are the coefficients for ramping up and down, respectively. This limits the pressure change to a maximum or minimum from $P_{i,t,prev}$, the plenum pressure used in the previous step. The total temperature does not use a ramping function, so $T_{i,t,plenum} = T_{i,t,cmd}$. The plenum boundary conditions are set using the total pressure and temperature ratios with respect to the reference quantities:

$$PR_i = P_{i,t,plenum}/P_{i,ref} \tag{26}$$

$$TR_i = T_{i,t,plenum}/T_{i,ref} \tag{27}$$

Careful accounting of the propulsive effects must be made between the trajectory propagator and flow solver. The pressure forces and momentum transfer across the plenum are calculated in the trajectory propagator, and the remaining propulsive forces and interaction are captured through the flow solver.

4. Aerodynamic Coefficients

FUN3D provides the axial, side, and normal force coefficients C_A, C_Y, and C_N in the Body frame that POST2 expects, so no transformation is needed for the force coefficients. The moment coefficients are calculated in the Body Reference frame, so these are transformed into the Body frame using Equation 28:

$$[C_{LL} \quad C_M \quad C_{LN}]_B^T = \boldsymbol{R}_{BF}[C_{LL} \quad C_M \quad C_{LN}]_F^T \tag{28}$$

The coefficients must be corrected to the actual freestream conditions by multiplying by Q/Q_{ref} where Q is the current dynamic pressure and Q_{ref} is the reference dynamic pressure used in FUN3D. FUN3D calculates the coefficients about the center of gravity, so if this point is used as the aerodynamic reference point in POST2, no further transformation is necessary.

5. State Interpolation for Larger POST2 Time Steps

On the computing clusters used to develop test this framework, a POST2 run takes about 0.01% of the time needed for a FUN3D run. For this system, running both POST2 and FUN3D at the same time step is reasonable, as the time taken for FUN3D to run is significantly larger than the time spent waiting on POST2 to conclude. As computational power increases, the flow solver (the prarllelizable part of the model) takes relatively less time compared to the trajectory propagator. To ameliorate this issue, the trajectory propagator can be run at a larger time step — an integer multiple of Δt_{fs} — while still resolving the flight dynamics. This change can yield practical benefits for high-powered computing as the net wait time for calculating the equations of motion and communicating the results are reduced. However, this requires interpolation between the vehicle states $\bar{X}_a = \bar{X}(t)$ and $\bar{X}_b = \bar{X}(t + \Delta t)$ at intervals of Δt_{fs} to provide grid motion updates to the flow solver. In order to preserve flow stability, these interpolations must be C^2 continuous; that is, the linear and angular acceleration must be continuous across steps.

Because translations operate in Euclidian space, the interpolation of the linear state components can be applied on a component-level basis. Within each interpolation step, the i^{th} component of the interpolated position $\bar{r}_{int,O}$ is assumed to have the form

$$x_{int,i,O} = c_{i,0} + c_{i,1}u + c_{i,2}u^2 + c_{i,3}u^3 \tag{29}$$

where $u = 0..1$ and c_0-c_3 are constants, found by solving the boundary problems:

$$\bar{r}_{int,O}(u=0) = \bar{r}_{a,O}, \quad \frac{d\bar{r}_{int,O}}{dt}(u=0) = \bar{v}_{a,O}, \quad \bar{r}_{int,O}(u=1) = \bar{r}_{b,O}, \quad \frac{d\bar{r}_{int,O}}{dt}(u=1) = \bar{v}_{b,O} \tag{30}$$

The Euler angles do not operate in Euclidian space, so the rotation cannot be interpolated at the component level. Therefore, the interpolated orientation will be found by operating with quaternions, since these will always produce an orthonormal rotation. Quaternion interpolation is commonly performed in the field of computer graphics [40], and as such, a variety of methods are available. A 3rd-order Bézier curve with Bernstein basis was selected for its ability to match the angular velocities at the start and end of the interpolation without any *a priori* knowledge of the interstitial state [41]. The curve takes the form

$$\bar{q}(u) = \bar{q}_0 \prod_{i=1}^{3} \exp(\tilde{\omega}_i \tilde{\beta}_{i,3}(u)) \tag{31}$$

where

$$\tilde{\beta}_{i,3}(u) = \sum_{j=i}^{3} \binom{3}{i} (1-u)^{3-i} u^i, \quad \tilde{\omega}_i = \log(\bar{q}_{i-1}^{-1} \bar{q}_i) \tag{32}$$

The values of the four control points $\bar{q}_0$ through $\bar{q}_3$ must be found. The endpoints are given by $\bar{q}_0 = \bar{q}_a$ and $\bar{q}_3 = \bar{q}_b$, and the midpoints are calculated by propagating the orientation forwards and backwards by a third of the POST2 time step:

$$\bar{q}_1 = \bar{q}_a + \frac{d\bar{q}_a}{dt}\frac{\Delta t}{3}, \quad \bar{q}_2 = \bar{q}_b - \frac{d\bar{q}_b}{dt}\frac{\Delta t}{3} \tag{33}$$

where

$$\frac{d\bar{q}_i}{dt} = 0.5 \begin{bmatrix} 0 \\ \bar{\omega}_i \end{bmatrix} \bar{q}_i \tag{34}$$

is found using the angular velocity at the endpoints. Within the framework, the rotation matrices $\boldsymbol{R}_{OF}$ and angular velocities $\omega_{rel,F}$ at the endpoints are converted to quaternion space, the interpolation is applied, and the results are converted back.

IV. Cross-Code Verification

Verification of the framework was performed by comparing its performance to an existing CFD-RBD environment, a 6DOF library that can be compiled with FUN3D [24, 33]. Cross-code verification against the 6DOF Library was performed by running a simulation with the same vehicle, initial conditions, flow solver settings, and initial flow history. The only differences were the trajectory propagator and the method of communication between the disciplinary codes.

The POST2 framework was set up in order to approximate the FUN3D 6DOF Library as closely as possible. The 6DOF Library uses a "flat Earth" model with a uniform gravitational field and inertial vertical frame. The flat Earth model is approximated in POST2 by simulating the vehicle around a non-rotating planet with a radius of 10^9 m and

a gravitational constant chosen to keep the sea-level gravitational acceleration equal to the standard acceleration of gravity. The 6DOF Library also calculates forces and moments using the flow solver reference quantities for density and speed, which are constant during the simulation. Therefore, constant sea-level atmospheric conditions were enforced within POST2. The 6DOF Library uses the same modification to the time integration scheme as the FUN3D-POST2 framework, so no modification was necessary in that regard.

The Army-Navy Finner (ANF) projectile was selected as the vehicle for the verification activity due to its common usage for aerodynamic research in both physical [42, 43] and computational [44, 45] experiments. The dimensions of the vehicle are shown in Figure 5, and the mass properties are listed in Table 1.

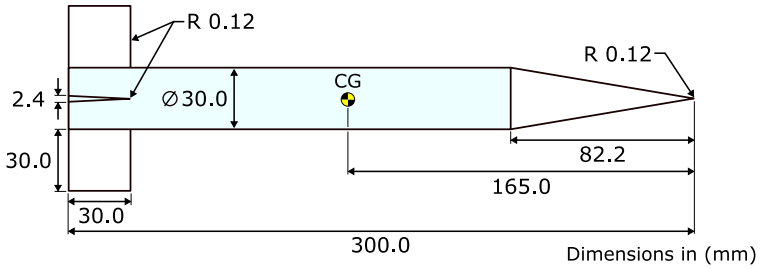


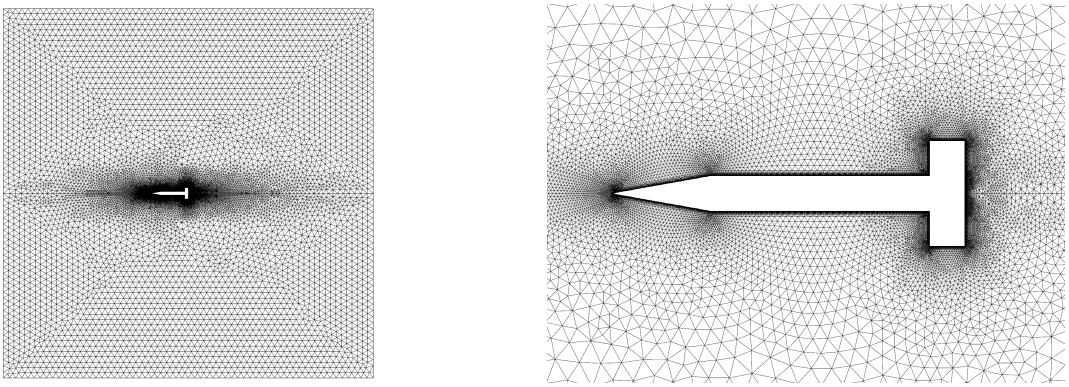
Fig. 5 Army-Navy finner projectile dimensions.

Table 1 ANF mass properties.

m, kg	I_{xx}, kg·m^2	I_{yy} and I_{zz}, kg·m^2
1.588	$1.92526 \cdot 10^{-4}$	$9.87035 \cdot 10^{-3}$

A. Grid Generation

Details of the grid generation are described in Reference [46], a summary of which is presented below. An unstructured, mixed-element grid was constructed for the ANF in Pointwise®V18.0R2. The surface and volume were generated for a quarter of the vehicle, then mirrored into a full volume to ensure symmetry.

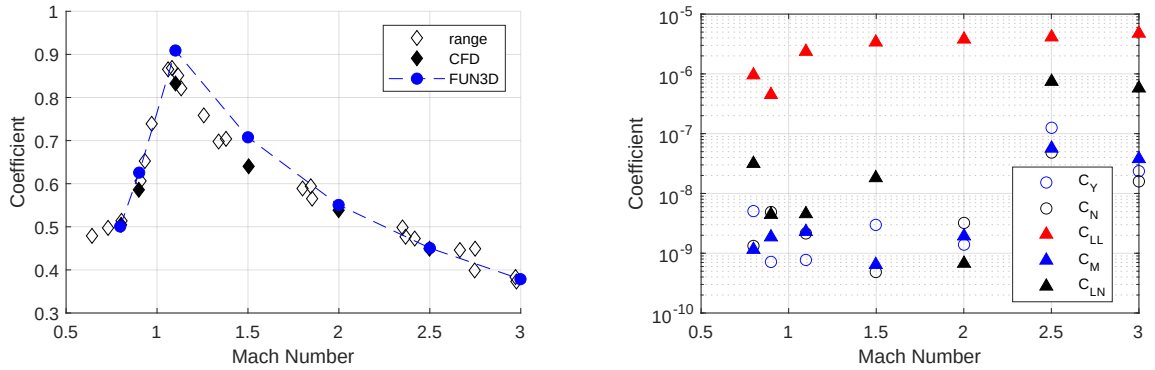


(a) Full grid **(b) Grid detail**

Fig. 6 ANF computational grid

Validation of the drag coefficient was performed against a set of ballistic range tests conducted by Dupuis and Hathaway [43, 47], as well as CFD experiments performed by Sahu and Heavey [44]. The comparison of axial results is shown in Figure 7a, and results for the off-axis coefficients are shown in Figure 7b. The results for the axial coefficient are a close match to both the physical and computational experiments, especially in the high supersonic regime. The roll

moment coefficient is resolved to less than 10^{-5}, and all other components are resolved to less than 10^{-6}. This precision was judged to be sufficient for the use of this grid in 6DOF simulations.



(a) Axial force coefficient vs Mach, $\alpha = 0°$. **(b) Magnitude of off-axis residuals for $\alpha = 0°$.**

Fig. 7 ANF Grid Validation Results[46]

B. Startup Procedure

The trajectory propagator and flow solver must be initialized with the vehicle state at the beginning of the simulation. While initializing the state variables for the rigid body via the input files is trivial, initializing the flow field is more complex. Since the coupled simulation uses the aerodynamic force and moment coefficients to update the 6DOF equations of motion, these coefficients must be converged at the start of the coupled simulation. Otherwise, numerical instability could cause non-physical forces and moments to be applied to the vehicle, affecting the trajectory or causing the simulation to fail.

In addition to meeting the threshold for convergence, the flow field must be accurate to the desired intial conditions. However, while initializing the trajectory propagator with the vehicle and control system state is sufficient, this does not ensure that the flow field matches. Initializing from a static CFD solution would result in a flow history that does not capture the unsteady behavior that presumably leads to the desired initial conditions. For example, consider the flow field around a vehicle at constant speed and slip angles, which would be different from one that was at a nonzero angluar velocity or was accelerating.

Starting from the desired initial conditions, functions are created for each state component, either fitted from experimental data or created from scratch to match the values and first derivatives of the initial conditions. These functions are then propagated backwards in time for a sufficient number of steps to allow the time-accurate simulation to converge. At the start of this time, a new set of conditions are used to run a static simulation. This flow field is used to initialize a time-accurate run with forced motion, which results in the flow field that is consistent with the desired, unsteady initial conditions. An example of these steps is shown for the Euler angles in Figure 8.

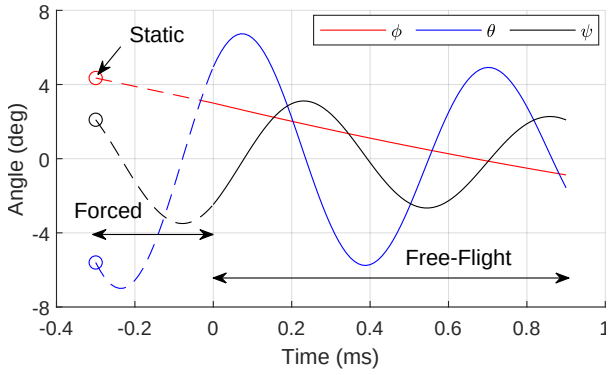
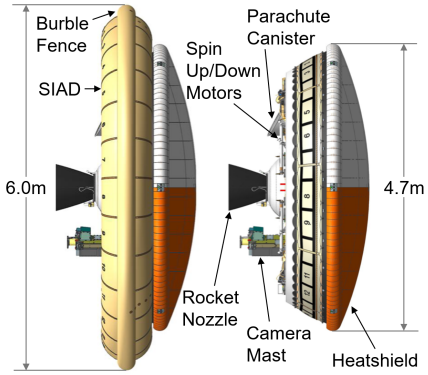


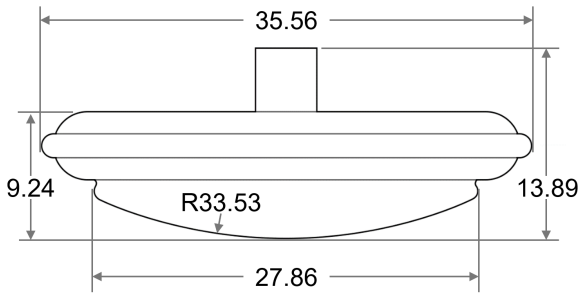
Fig. 8 Trajectory initialization example

V. Validation

Validation of the framework was performed by replicating a series of ballistic range tests for an entry vehicle scale model. The Supersonic Inflatable Aerodynamic Decelerator (SIAD) is an inflatable torus designed to increase the drag area of an entry vehicle [48]. Deployed and stowed configurations of the flight test vehicle are shown in Figure 9a. Scale models of a SIAD-equipped vehicle were flown in ballistic range shots at the NASA Ames Hypervelocity Free-Flight Aerodynamics Facility [49]. The model dimensions are shown in Figure 9b. The model would impact a sheet of paper as it entered the measurement section of the range, inducing an oscillation in pitch and yaw. Reconstruction of the trajectory demonstrated that some of the shots had significant nonzero initial roll rates [50]. The unsteady initial conditions make these tests a suitable validation case for the framework.



(a) Full-scale vehicle [51]



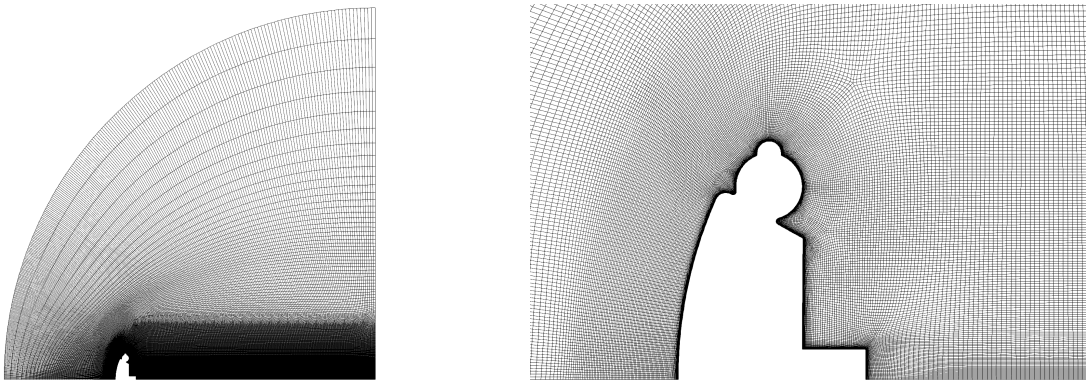
(b) Ballistic range model dimensions (mm) [50]

Fig. 9 SIAD configuration

Table 2 SIAD mass properties [49]

Mass, g	x_{cg}/d	I_{xx}, g·cm^2	I_{yy} and I_{zz}, g·cm^2
45.93	0.161	3.67	2.11

A computational mesh for the SIAD model, shown in Figure 10, was created by Brock et al. to perform replications of the ballistic range tests in US3D [23]. This grid was provided for use in the POST2-FUN3D framework validation. The grid density was chosen to ensure a y^+ of less than 1.0 at the vehicle surface and in the wake in order to properly resolve the bluff body flow. The grid was created with quarter-symmetry, then mirrored about the x-y and x-z planes, for a total size of around 22 million nodes.



(a) Quarter-symmetry

(b) Grid detail

Fig. 10 SIAD computational grid[39]

VI. Results

A. Cross-Code Verification

The ANF projectile was simulated in a ballistic drop at Mach 2.0 at standard sea-level conditions, with zero initial slip angles and angular rates. The flow solver used a k-omega turbulence model [52] with a low-diffusion flux-splitting scheme. The boundary conditions consisted of a viscous, no-slip condition on the projectile surface and a Riemann invariant farfield at the outside surfaces of the grid. Both simulations were performed on the Aerospace Systems Design Laboratory computing cluster within the Partnership for an Advanced Computing Environment at the Georgia Institute of Technology [53]. The simulations were run with $\Delta t = 4.4079 \times 10^{-4}$s for 200 steps, or about 44ms of simulation time.

A comparison of the framework ('POST') and FUN3D 6DOF Library ('Library') results for the ballistic drop are shown below in Figure 11. The results shown in Figures 11a and 11b describe a ballistic trajectory with pitch oscillation, which is the expected behavior. Figures 11c and 11d show that nonzero but small lateral motion develops due to computational precision. The error in each of the displacement components, shown in Figure 11e, is small compared to the overall motion of the vehicle, and each is of about the same order of mangitude. The orientation error plotted in Figure 11f shows that the error in the roll angle is the largest, which is expected from the behavior identified during grid validation.

Figure 12 plots the magnitude of the displacement and orientation errors by simulation step. Both simulations are started from the same initial conditions, so the error is initially zero. After the first step, the error in all components is on the order of 10^{-16}. The displacement error magnitude rises more sharply than the orientation, reaching between 10^{-8} and 10^{-7} for the duration of the simulation. The orientation error magnitude increases more slowly but reaches between 10^{-7} and 10^{-4}. Based on this rate of error growth over time, the cross-code verification was judged to be successful.

B. Validation

Three SIAD ballistic range shots were replicated, with initial conditions spanning Mach 2 to 4 and with a total angle of attack amplitude of up to 18.4°. One of the replicated shots, Shot 2643, is presented below, and further detail can be found in Reference [50]. The initial conditions for the recreation are listed in Table 3. Shot 2643 had the highest calculated initial roll rate, at 1890°/s.

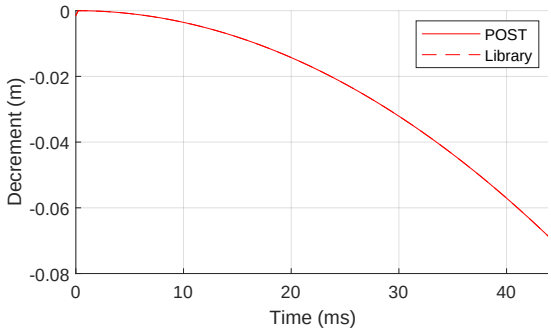
Table 3 SIAD ballistic range Shot 2643 recreation initial conditions

Pres., kPa	Temp., K	v_x, m/s	v_y, m/s	v_z, m/s	ϕ, °	θ, °	ψ, °	ω_x, °/s	ω_y, °/s	ω_z, °/s
19.87	294.2	1188.9	-0.75	-0.11	0	-1.64	0.74	-1890	6100	-1030

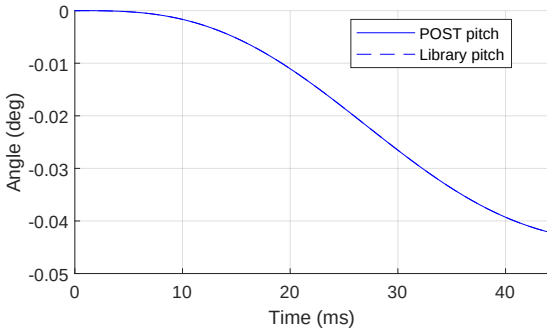
The flow solver modeled fully turbulent flow with a Spalart-Allmaras-based DES turbulence model [54]. The flux limiter was a stencil-based van Albada flux scheme [55], and a dissipative low-diffusion flux splitting scheme was used. The simulation used a Δt and Δt_{fs} of 1.4955×10^{-5}s and was executed for a duration of about 20ms.

The results of the ballistic range replication are shown below in Figure 13. The x-axis displacement was within 0.0656% of the experimental displacement at the end of the shot. The simulation captures the slight oscillatory behavior of the lateral displacment components, with the y-axis displacement almost entirely within the experimental measurement uncertainty and the z-axis displacement on the same order of magnitude.

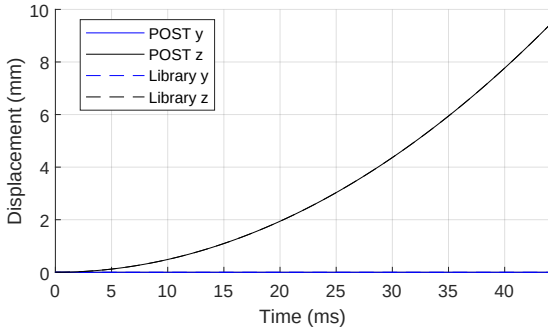
The pitch and yaw behavior matches the experimental measurements, including the exchange in oscillation amplitude as the roll angle changes. The simulated roll angle is plotted in Figure 13g against the calculated experimental roll angle, since roll was not measured directly during the experiment (note that uncertainty propagation results in uncertainty bounds that are sensitive to the total angle of attack). The roll angle error is plotted in Figure 13h against the root-mean-square of the calculated uncertainty. Excluding the outliers, the simulation matched the calculated roll angle to within 25°. The SIAD simulations for all three shots in the POST2-FUN3D framework yielded a similar accuracy to the results of Brock et al.[23, 50].



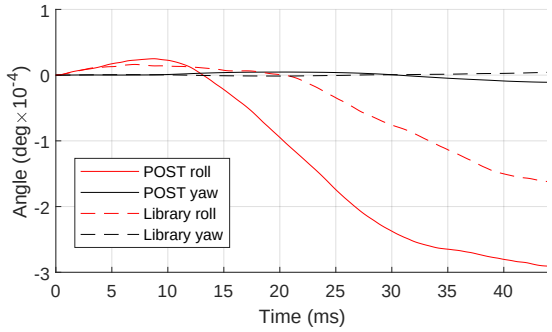
(a) x-axis decrement comparison



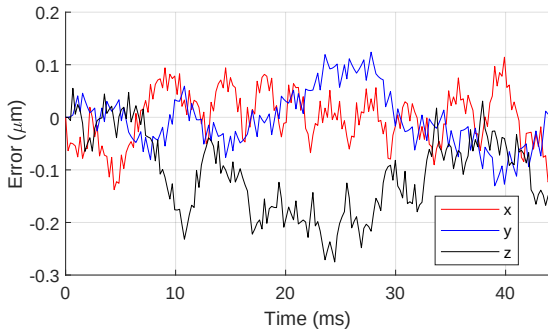
(b) Pitch angle comparison



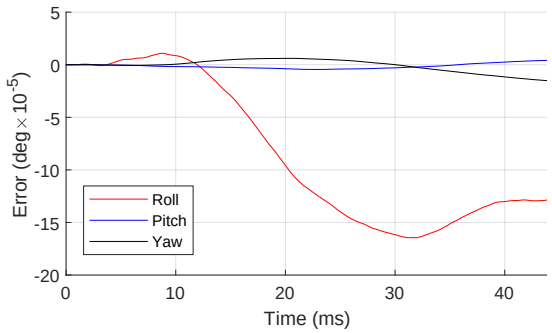
(c) Lateral displacement comparison



(d) Roll, yaw angle comparison

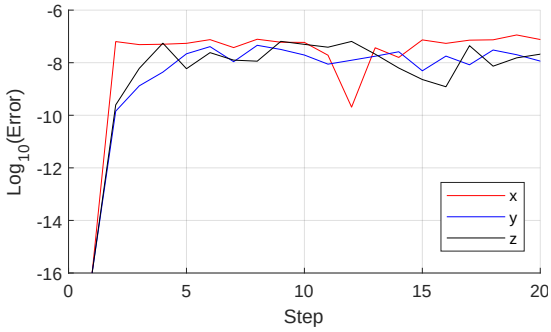


(e) Displacement error

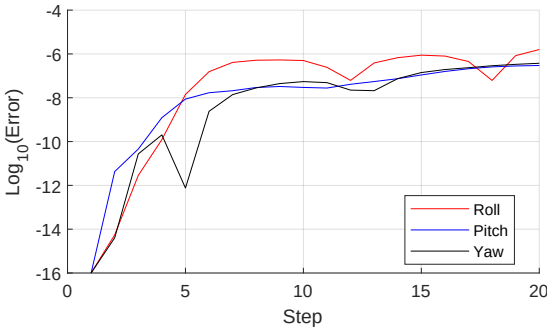


(f) Orientation error

Fig. 11 ANF cross-code verification results

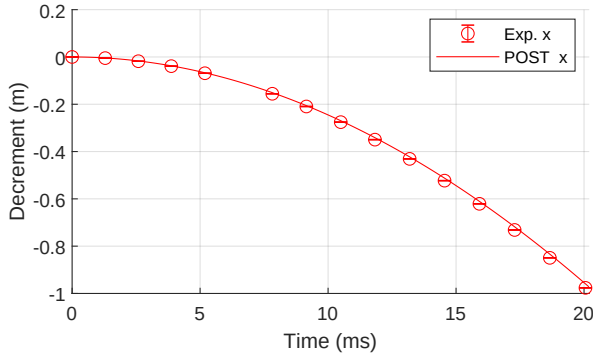


(a) Displacment

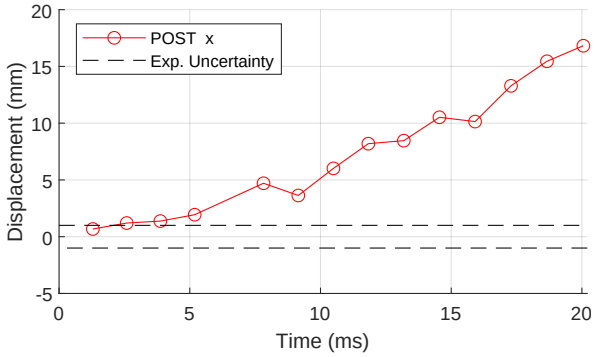


(b) Orientation

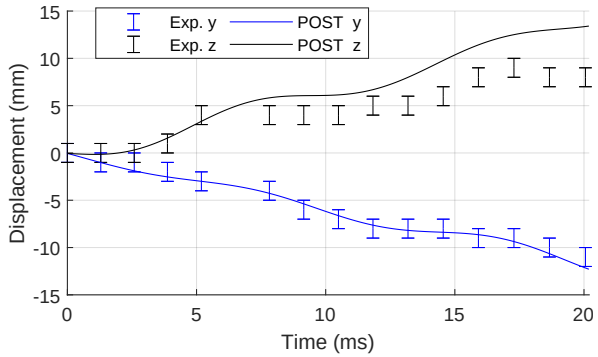
Fig. 12 ANF cross-code verification error growth



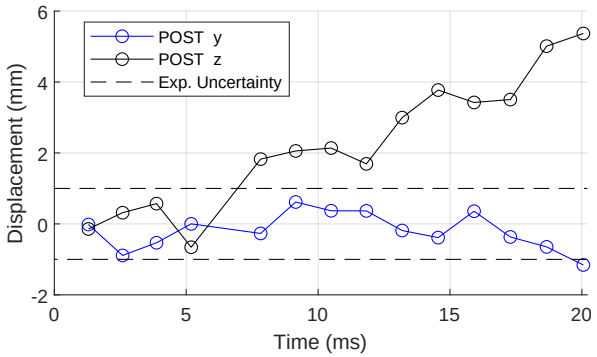
(a) x-axis decrement



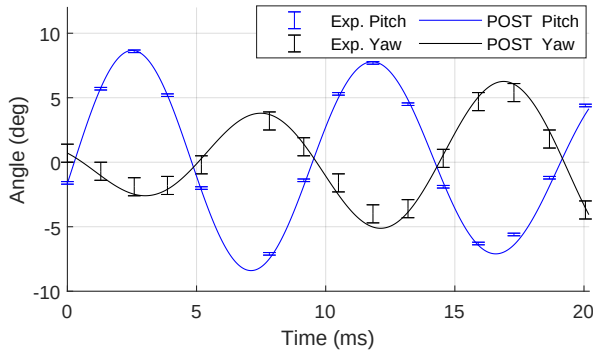
(b) x-axis decrement error



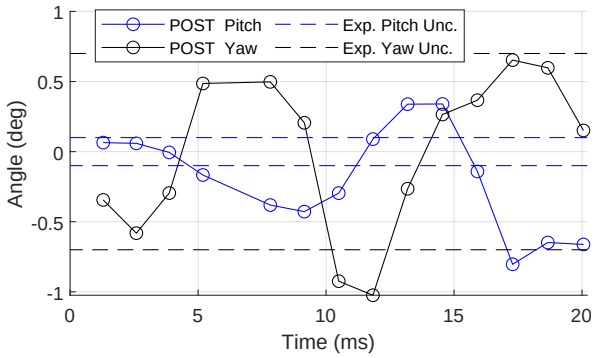
(c) Lateral displacement



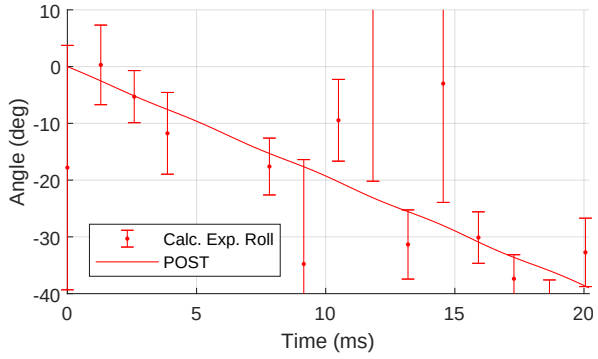
(d) Lateral displacement error



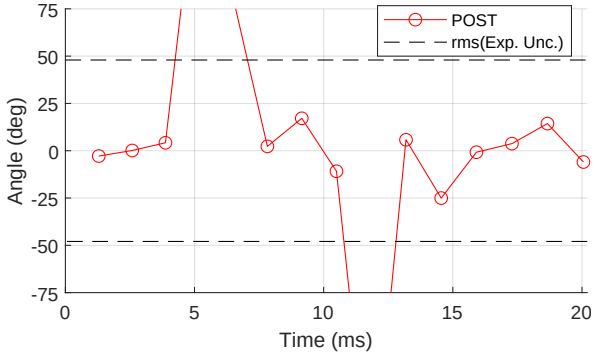
(e) Pitch, yaw angle



(f) Pitch, yaw angle error



(g) Roll angle



(h) Roll angle error

Fig. 13 SIAD ballistic range test recreation[39]

VII. Conclusion

The POST2-FUN3D framework is a CFD-RBD model that provides expanded capability for multidisciplinary/multiphysics simulation. By linking FUN3D to POST2, a state-of-the-art trajectory propagator, designers will be able to include other disciplinary models that are constructed for POST2, including GN&C and propulsion.

The cross-code verification of the framework against the FUN3D 6DOF Library was successful, with the error in the trajectory components matching to within the order of 10^{-4}. Free-flight simulations in the framework have been successfully validated against both experimental data and independent CFD-RBD simulations. Further research is underway to incorporate open- and closed-loop control with RCS jets for an entry vehicle [39].

VIII. Acknowledgements

The authors wish to thank the NASA Engineering and Safety Center and the Comprehensive Digital Transformation program for sponsoring this work. Additionally, we would like to thank Joseph M. Brock, Eric C. Stern, Michael C. Wilder, Paul V. Tartabini, Anthony Williams, Robert T. Biedron, Kevin Jacobson, Li Wang, Dan Murri, and David Schuster for their support of this work. This research was also supported in part through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment at the Georgia Institute of Technology, Atlanta, Georgia, USA.

References

[1] Lugo, R. A., Shidner, J. D., Powell, R. W., Marsh, S. M., Hoffman, J. A., Litton, D. K., and Schmitt, T. L., "Launch vehicle ascent trajectory simulation using the Program to Optimize Simulated Trajectories II (POST2)," 2017.

[2] Dutta, S., and Way, D. W., "Comparison of the effects of velocity and range triggers on trajectory dispersions for the Mars 2020 mission," *AIAA Atmospheric Flight Mechanics Conference*, 2017, p. 0245.

[3] Atkins, B. M., and Queen, E. M., "Internal moving mass actuator control for mars entry guidance," *Journal of Spacecraft and Rockets*, Vol. 52, No. 5, 2015, pp. 1294–1310.

[4] Striepe, S. A., Way, D., Dwyer, A., and Balaram, J., "Mars science laboratory simulations for entry, descent, and landing," *Journal of Spacecraft and Rockets*, Vol. 43, No. 2, 2006, pp. 311–323.

[5] Pinier, J. T., Bennett, D. W., Erickson, G. E., Favaregh, N. M., Houlden, H. P., Tomek, W. G., and Blevins, J. A., "Space Launch System Ascent Static Aerodynamics Database Development," *52nd Aerospace Sciences Meeting*, 2014, p. 1254.

[6] Dyakonov, A., Schoenenberger, M., and Van Norman, J., "Hypersonic and supersonic static aerodynamics of Mars science laboratory entry vehicle," *43rd AIAA Thermophysics Conference*, 2012, p. 2999.

[7] Hall, L. H., Eppard, W., Applebaum, M., and Purinton, D., "Modeling and Simulation Techniques for the NASA SLS Service Module Panel Separation Event; From Loosely-Coupled Euler to Full-Coupled 6-DOF, Time-Accurate, Navier Stokes Methodologies," *AIAA Scitech 2019 Forum*, 2019, p. 1843.

[8] Murman, S. M., Diosady, L. T., and Blonigan, P. J., "Comparison of Transonic Buffet Simulations with Unsteady PSP Measurements for a Hammerhead Payload Fairing," *55th AIAA Aerospace Sciences Meeting*, 2017, p. 1404.

[9] Brune, A. J., Hosder, S., and Edquist, K. T., "Uncertainty analysis of fluid-structure interaction of a deformable hypersonic inflatable aerodynamic decelerator," *Journal of Spacecraft and Rockets*, Vol. 53, No. 4, 2016, pp. 654–668.

[10] Dyakonov, A., Schoenenberger, M., Scallion, W., Van Norman, J., Novak, L., and Tang, C., "Aerodynamic interference due to MSL reaction control system," *41st AIAA Thermophysics conference*, 2009, p. 3915.

[11] Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences: NASA," Tech. rep., CR-2014-218178Washington, DC: NASA, 2014.

[12] Cenko, A., "Store separation lessons learned during the last 30 years," Tech. rep., NAVAL AIR SYSTEMS COMMAND PATUXENT RIVER MD, 2010.

[13] Meakin, R., "Computations of the unsteady flow about a generic wing/pylon/finned-store configuration," *Astrodynamics Conference*, 1992, p. 4568.

[14] Kokes, J., Costello, M., and Sahu, J., "Generating an Aerodynamic Model for Projectile Flight Simulation Using Unsteady Time Accurate Computational Fluid Dynamic Results," *Computational Ballistics III*, Vol. 45, 2007, p. 11131.

[15] Costello, M., Gatto, S., and Sahu, J., "Using CFD/RBD Results to Generate Aerodynamic Models for Projectile Flight Simulation," *AIAA Atmospheric Flight Mechanics Conference*, 2007. Talks about complexity of initialization.

[16] Montalvo, C., and Costello, M., "Estimation of Projectile Aerodynamic Coefficients Using Coupled CFD/RBD Simulation Results," *AIAA Atmospheric Flight Mechanics Conference*, American Institute of Aeronautics and Astronautics, 2010. doi:10.2514/6.2010-8249, continuation of costello2007projectile.

[17] Sahu, J., "Numerical Computations of Dynamic Derivatives of a Finned Projectile Using a Time-Accurate CFD Method," *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2007, p. 6581.

[18] Sahu, J., and Fresconi, F., "Flight behaviors of a complex projectile using a coupled CFD-based simulation technique: open-loop control," *54th AIAA Aerospace Sciences Meeting*, 2016, p. 2025.

[19] Sahu, J., and Fresconi, F., "Flight behaviors of a complex projectile using a coupled CFD-based simulation technique: closed-loop control," *34th AIAA Applied Aerodynamics Conference*, 2016, p. 4332.

[20] Murman, S., Aftosmis, M., and Berger, M., "Simulations of 6-DOF motion with a Cartesian Method," *41st Aerospace Sciences Meeting and Exhibit*, 2003, p. 1246.

[21] Stern, E., Gidzak, V., and Candler, G., "Estimation of Dynamic Stability Coefficients for Aerodynamic Decelerators Using CFD," *30th AIAA Applied Aerodynamics Conference*, American Institute of Aeronautics and Astronautics, 2012. doi:10.2514/6.2012-3225.

[22] Stern, E., Schwing, A., Brock, J. M., and Schoenenberger, M., "Dynamic CFD Simulations of the MEADS II Ballistic Range Test Model," *AIAA Atmospheric Flight Mechanics Conference*, 2016, p. 3243.

[23] Brock, J. M., Stern, E. C., and Wilder, M. C., "Computational Fluid Dynamics Simulations of Supersonic Inflatable Aerodynamic Decelerator Ballistic Range Tests," *Journal of Spacecraft and Rockets*, Vol. 56, No. 2, 2019, pp. 526–535. doi:10.2514/1.a34208.

[24] Koomullil, R., and Prewitt, N., "A Library Based Approach for Rigid Body Dynamics Simulation," *18th AIAA Computational Fluid Dynamics Conference*, 2007, p. 4476.

[25] Schuster, D. M., "CFD 2030 Grand Challenge: CFD-in-the-Loop Monte Carlo Flight Simulation for Space Vehicle Design," *AIAA Scitech 2021 Forum*, 2021, p. 0957.

[26] Etkin, B., *Dynamics of Atmospheric Flight*, Courier Corporation, 2012.

[27] Butcher, J. C., *Numerical Methods for Ordinary Differential Equations*, John Wiley & Sons, 2016.

[28] Biedron, R., and Thomas, J., "Recent Enhancements to the FUN3D Flow Solver for Moving-Mesh Applications," *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, 2009, p. 1360.

[29] Anderson, W. K., and Bonhaus, D. L., "An implicit upwind algorithm for computing turbulent flows on unstructured grids," *Computers & Fluids*, Vol. 23, No. 1, 1994, pp. 1–21.

[30] Biedron, R., Vatsa, V., and Atkins, H., "Simulation of Unsteady Flows Using an Unstructured Navier-Stokes Solver on Moving and Stationary Grids," *23rd AIAA Applied Aerodynamics Conference*, 2005, p. 5093.

[31] Matthies, H. G., and Steindorf, J., "Partitioned but strongly coupled iteration schemes for nonlinear fluid–structure interaction," *Computers & structures*, Vol. 80, No. 27-30, 2002, pp. 1991–1999.

[32] Warner, J., Niemoeller, S. C., Morrill, L., Bomarito, G., Leser, P., Leser, W., Williams, R. A., and Dutta, S., "Multi-Model Monte Carlo Estimators for Trajectory Simulation," *AIAA Scitech 2021 Forum*, 2021, p. 0761.

[33] Biedron, R. T., Carlson, J. R., Derlaga, J. M., Gnoffo, P. A., Hammond, D. P., Jones, W. T., Kleb, B., Lee-Rausch, E. M., Nielsen, E. J., Park, M. A., et al., *FUN3D Manual: 13.6*, 2019.

[34] Alexandrov, N., Atkins, H., Bibb, K., Biedron, R., Carpenter, M., Gnoffo, P., Hammond, D., Jones, W., Kleb, W., and Lee-Rausch, E., "Team software development for aerothermodynamic and aerodynamic analysis and design," 2003.

[35] Dyakonov, A. A., Glass, C. E., Desai, P. N., and Van Norman, J. W., "Analysis of Effectiveness of Phoenix Entry Reaction Control System," *Journal of Spacecraft and Rockets*, Vol. 48, No. 5, 2011, pp. 746–755.

[36] Striepe, S., Powell, R., Desai, P., Queen, E., Way, D., Prince, J., Cianciolo, A., Davis, J., Litton, D., Maddock, R., Shidner, J. D., Winski, R. G., O'Keefe, S. A., Bowes, A. G., Aguirre, J. T., Garrison, C. A., Hoffman, J. A., Olds, A. D., Dutta, S., Brauer, G. L., Engel, M. C., and Marsh, S. M., *Program To Optimize Simulated Trajectories II (POST2): Utilization Manual*, 2015.

[37] Kornfeld, R. P., Prakash, R., Devereaux, A. S., Greco, M. E., Harmon, C. C., and Kipp, D. M., "Verification and Validation of the Mars Science Laboratory/Curiosity Rover Entry, Descent, and Landing System," *Journal of spacecraft and rockets*, Vol. 51, No. 4, 2014, pp. 1251–1269.

[38] White, J., Bowes, A. L., Dutta, S., Ivanov, M. C., and Queen, E. M., "LDSD POST2 Modeling Enhancements in Support of SFDT-2 Flight Operations," 2016.

[39] Petitgenet, V., Hickey, A., Robertson, B., and Mavris, D., "Actuating Reaction Control System Jets in a Time-Accurate Supersonic Cross Flow," *AIAA Scitech 2022 Forum (submitted for publication)*, 2022.

[40] Shoemake, K., "Animating rotation with quaternion curves," *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985, pp. 245–254.

[41] Myoung-Jun Kim, S. Y. S., Myung-Soo Kim, "A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives," *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, edited by editor, ACM, 1985. doi:10.445/218380.218486.

[42] MacAllister, L., "The Aerodynamic Properties of a Simple Non Rolling Finned Cone-Cylinder Configuration Between Mach Numbers 1.0 and 2.5," Tech. rep., Army Ballistic Research Lab, Aberdeen Proving Ground, MD, 1955.

[43] Dupuis, A. D., and Hathaway, W., "Aeroballistic Range Tests of the Basic Finner Reference Projectile at Supersonic Velocities," Tech. rep., Defence Research Establishment Valcartier (Québec), 1997.

[44] Sahu, J., and Heavey, K. R., "Parallel CFD Computations of Projectile Aerodynamics with a Flow Control Mechanism," *Computers & Fluids*, Vol. 88, 2013, pp. 678–687.

[45] Dykes, J., Montalvo, C., Costello, M., and Sahu, J., "Use of Microspoilers for Control of Finned Projectiles," *Journal of Spacecraft and Rockets*, Vol. 49, No. 6, 2012, pp. 1131–1140.

[46] Ernst, Z. J., Hiller, B. R., Johnson, C. L., Robertson, B. E., and Mavris, D. N., "Coupling Computational Fluid Dynamics with 6DOF Rigid Body Dynamics for Unsteady, Accelerated Flow Simulations," *2018 AIAA Atmospheric Flight Mechanics Conference*, 2018, p. 0291.

[47] Dupuis, A., "Aeroballistic Range and Wind Tunnel Tests of the Basic Finner Reference Projectile From Subsonic to High Supersonic Velocities," *Defense R&D Canada, Technical Memorandum TM 2002-136*, 2002.

[48] Giersch, L., Rivellini, T., Clark, I. G., Sandy, C., Sharpe, G., Shook, L. S., Ware, J. S., Welch, J., Mollura, J., and Dixon, M., "SIAD-R: A supersonic inflatable aerodynamic decelerator for robotic missions to mars," *AIAA Aerodynamic Decelerator Systems (ADS) Conference*, 2013, p. 1327.

[49] Wilder, M. C., Brown, J. D., Bogdanoff, D. W., Yates, L. A., Dyakonov, A. A., Clark, I. G., and Grinstead, J. H., "Aerodynamic Coefficients from Aeroballistic Range Testing of Deployed-and Stowed-SIAD SFDT Models," 2017.

[50] Ernst, Z. J., Hickey, A., Robertson, B., and Mavris, D., "Impact of Roll Rate on Free-Flight CFD Modeling of Entry Vehicles," *Journal of Spacecraft and Rockets (submitted for publication)*, 2021.

[51] Cook, B. T., Blando, G., Kennett, A., Heydt, M. V. D., Wolff, J. L., and Yerdon, M., "High Altitude Supersonic Decelerator Test Vehicle," *AIAA Aerodynamic Decelerator Systems (ADS) Conference*, American Institute of Aeronautics and Astronautics, 2013. doi:10.2514/6.2013-1353.

[52] Wilcox, D. C., "Formulation of the kw turbulence model revisited," *AIAA journal*, Vol. 46, No. 11, 2008, pp. 2823–2838.

[53] PACE, *Partnership for an Advanced Computing Environment (PACE)*, 2022. URL `http://www.pace.gatech.edu`.

[54] Spalart, P. R., "Comments on the feasibility of LES for wings, and on a hybrid RANS/LES approach," *Proceedings of first AFOSR international conference on DNS/LES*, Greyden Press, 1997.

[55] Van Albada, G. D., Van Leer, B., and Roberts, W., "A comparative study of computational methods in cosmic gas dynamics," *Upwind and high-resolution schemes*, Springer, 1997, pp. 95–103.