AIAA-2018-1400

AIAA SciTech Forum, 8-12 January 2018, Kissimmee, Florida

Design of a modular monolithic implicit solver for multi-physics applications

Corentin Carton de Wiart*, Laslo T. Diosady†, Anirban Garai†, Nicholas Burgess†, Patrick Blonigan*, Dirk Ekelschot*, and Scott M. Murman‡

NASA Ames Research Center, Moffett Field, CA, USA

The design of a modular multi-physics high-order space-time finite-element framework is presented together with its extension to allow monolithic coupling of different physics. One of the main objectives of the framework is to perform efficient high-fidelity simulations of capsule/parachute systems. This problem requires simulating multiple physics including, but not limited to, the compressible Navier-Stokes equations, the dynamics of a moving body with mesh deformations and adaptation, the linear shell equations, non-reflective boundary conditions and wall modeling. The solver is based on high-order space-time finite element methods. Continuous, discontinuous and C^1-discontinuous Galerkin methods are implemented, allowing one to discretize various physical models. Tangent and adjoint sensitivity analysis are also targeted in order to conduct gradient-based optimization, error estimation, mesh adaptation, and flow control, adding another layer of complexity to the framework. The decisions made to tackle these challenges are presented. The discussion focuses first on the "single-physics" solver and later on its extension to the monolithic coupling of different physics. The implementation of different physics modules, relevant to the capsule/parachute system, are also presented. Finally, examples of coupled computations are presented, paving the way to the simulation of the full capsule/parachute system.

I. Introduction

Many engineering systems are inherently multi-disciplinary. Accurate modeling of such systems requires the coupling of multiple physical models, with potentially different levels of fidelity and different time scales. Examples include fluid-structure interaction (FSI), aero-thermal heating/conjugate heat transfer, chemically reacting flows, etc.[1] In this work, the main target applications is the high-fidelity simulation of capsule/parachute systems for atmospheric entry.[2] As shown on Figure 1, this application presents a significant modeling challenge due to the following features:

- high-Reynolds number flow about the capsule and the parachutes;
- high spatial and temporal resolution required for resolving both the parachute geometry and the turbulent flow;
- the interaction of the turbulent wake from the capsule with the thin and light-weight parachute;
- complex dynamics of the parachute fabric, which could also include the effects of porosity;
- dynamic system of multiple moving bodies with collisions and topology changes.

Many different physical phenomena are involved in this system and the physical models for each of these phenomena need to be tightly coupled in order to obtain a high-fidelity representation of the problem.

With increased complexity and fidelity of domain specific solvers, multi-physics simulation tools have generally been built based on modular frameworks.[1,3–11] Typically, numerical methods for such multi-physics

*USRA/NASA Postdoctoral Program Fellow.
†Science and Technology Corp.
‡NASA ARC. AIAA Member.

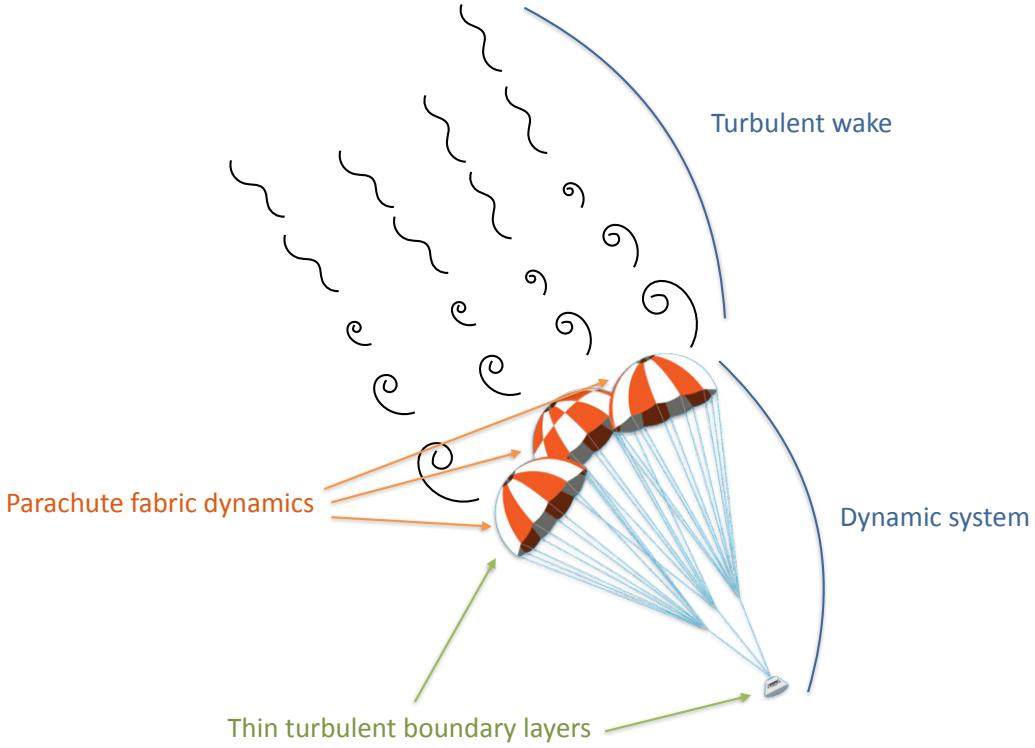


Figure 1. Sketching of the capsule/parachute system and the associated physics.

tools are based on partitioned schemes; separate solvers are developed for each discipline, leveraging expertise and existing software for each domain. Coupling of multiple disciplines is often performed as an after-thought, with coupling schemes which may be non-conservative, inaccurate or numerically unstable.[1] Nonetheless, such coupling methods have been successfully applied to simulate multi-disciplinary phenomenon where the coupling errors may be either insignificant due to the loosely coupled nature of the problem, or damped by the low-order numerical methods of the domain-specific solvers.

Alternatively, monolithic approaches solve the complete system of equations corresponding to the coupled multiphysics problem simulatenously.[1,3] As such, monolithic approaches may be designed to be conservative, higher-order and stable.[3,12] Despite these clear advantages, monolithic approaches are less widely used because they are seen as more expensive, less modular and more complex.[3] These concerns have predominantly been raised with legacy codes, for which there is significant inertia based on years of domain specific development.[1] Recent multi-physics codes have shown the possibility of both improved accuracy and efficiency while maintaining a modular framework,[3,13] though the initial development cost may be significant.

In our recent work, we have been developing a high-order space-time discontinuous-Galerkin (DG) finite-element method for high Reynolds number compressible turbulent flow simulations.[14–20] We have begun expanding the capability of our solver in order to perform scale-resolving simulations for a variety of problems of interest to NASA. We have shown the advantages of using higher-order methods for compressible turbulent flows[14,15,17] and our desire is to take advantage of higher-order methods for the simulation of the multi-physics problems. Furthermore, we wish to use tangent and adjoint sensitivity analysis to conduct gradient-based optimization, error estimation, mesh adaptation, and flow control on multi-physics problems with our solver.[19,21]

Higher-order methods can provide greater efficiency for simulations requiring high spatial and temporal resolution, allowing for solutions with fewer degrees of freedom and lower computational cost than traditional second-order computational fluid dynamics (CFD) methods. However, the reduced numerical stabilization present in higher-order schemes implies that special care needs to be taken in the development of numerical methods to suppress nonlinear instabilities.[17,22–26] Our numerical scheme for the fluid solver is based on a nonlinearly-stable space-time discontinuous-Galerkin method which has been used to perform turbulent flow

simulations up to 16th-order accuracy in space and time.[17] A space-time finite-element formulation can be naturally extended to moving-domain problems while guaranteeing satisfaction of the geometric conservation law (provided sufficient integration is used).[27] This has made the space-time DG formulation a natural choice for moving-domain and FSI simulations.[28–30] Due to the low numerical dissipation present in higher-order schemes, inconsistent coupling of the fluid and structure modules can lead to numerical instabilities and catastrophic failures of the simulation.[12,28,31] For this reason, a monolithic fully coupled space-time approach has been preferred to a partitioned method. This approach ensures discrete conservation, numerical stability and high-order accuracy on the coupled multi-physics problem.[12,28]

In this work, we discuss the extension of our existing fluid solver to a modular monolithic higher-order finite-element based multi-physics solver. We describe some of the challenges and strategies employed. In Section II we describe our existing software framework designed to handle a single physical model. Examples of some of the physical models and their discretizations are given in Section III. In Section IV we present the chosen coupling strategy. Some preliminary numerical results are presented in Section V. Finally, we discuss future perspectives in Section VI.

II. Single physics module

The existing higher-order space-time discontinuous-Galerkin compressible Navier-Stokes solver is the starting point of this work.[14,15,17] This solver is based on an efficient tensor-product finite-element formulation, a Jacobian-free Newton-Krylov solver and tensor-product preconditioners.[14,16] The solver has been validated up to 16th-order accuracy on simple test problems[15,17] and has been used for performing scale-resolving simulations on a variety of separated flows.[15,32]

The framework has been extended to allow for several finite-element discretizations (continuous-Galerkin/ discontinuous-Galerkin/C^1-discontinuous-Galerkin) to be used in combination with different sets of physics modules (compressible Navier-Stokes, elasticity, advection-diffusion, etc.). Examples of several physics modules are described in more detail in Section III. An object-oriented methodology is used, whereby main objects are responsible for

- defining the mesh and associated fields on the region;
- the finite-element discretization (CG/DG/C^1-DG);
- the non-linear and linear solvers;
- handling the primal, tangent and adjoint systems;
- the definition of the physics modules.

In particular, the finite-element discretization of each physical model involves evaluating integrals over the elements and faces of the mesh for computing residuals, linearized residuals, diagnostics etc. Each physics module implements an Application Programming Interface (API) for evaluating the integrands (fluxes, forces, outputs, etc.). The discretization modules implement the computation of the actual integrals by looping over appropriate elements and faces in the mesh, calling the mesh API to extract data at quadrature points and, in turn, calling the physics module APIs to evaluate the appropriate integrands. This modular and object-oriented implementation allows use of the same objects in different contexts (different physics or discretizations, primal or adjoint solve, etc.). The goal is that researchers can focus on their physics/model implementation without having to understand the back-end of the solver.

The modular implementation is further complicated by the requirement to support the ability to solve tangent and adjoint problems for sensitivity analysis.[19,21] The tangent and adjoint problems are linear systems of equations corresponding to a linearized physical model and its discrete transpose. In this approach, each physics module implements three versions of the integrands (fluxes, outputs functions, etc.) corresponding to primal, tangent and adjoint formulations. A common solver is used to converge primal, adjoint and tangent systems, with significant reuse of the discretization related modules. In particular, most operations in the residual evaluation are symmetric (i.e. apply basis functions - compute integrand - apply transpose of basis functions) such that the adjoint can be computed by simply transposing the primal integrand. Special care must be taken for certain non-symmetric operations.

The desire to compute the tangent and adjoint for simulations requires significant reading and writing of solution files to disk, as the primal solution for every time-slab must be saved in order to be able to

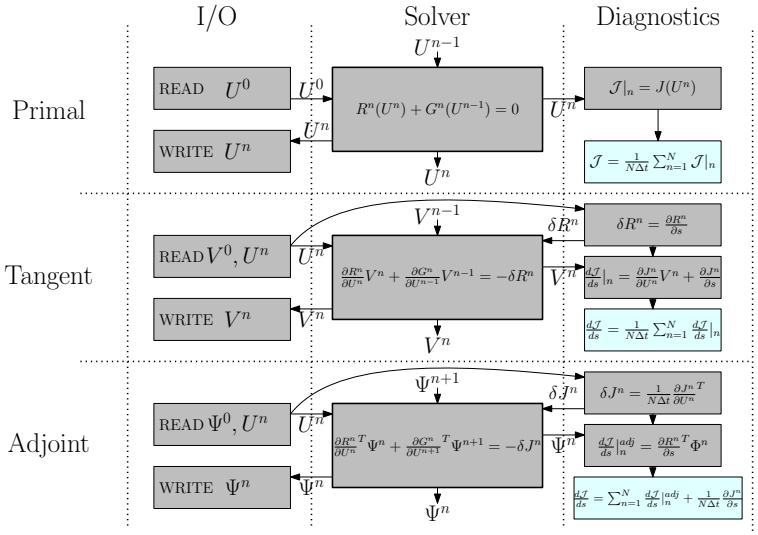


Figure 2. Flow charts for the primal, tangent, and adjoint solver. The primal solves the governing equations $R^n(U^n) + G^n(U^{n-1}) = 0$ for a primal solution U^n, where R^n and G^n are the residual and temporal source operators, respectively.[19] The primal diagnostics module computes an output quantity $\mathcal{J}|_n$ and its temporal mean $\mathcal{J}$. The tangent and adjoint solvers compute the tangent and adjoint solutions V^n and Ψ^n, respectively. The tangent and adjoint diagnostics modules compute the sensitivity of the temporal mean $\mathcal{J}$ to some input parameter(s) s.

linearize about this solution for the tangent and adjoint solves. Reading and writing to disk can be a significant bottleneck limiting the performance of the simulation tool. Additionally, parallel input/output (I/O) is known to scale poorly with the number of processors so it can be problematic for massively parallel simulations. In order to overcome these bottlenecks, asynchronous I/O is performed on a separate set of dedicated cores. A similar approach is used for the computation of diagnostics (cut-planes, integrals, averaging, etc.) which are also computed on a separate set of cores. Figure 2 depicts the process flow between I/O, solver, and diagnostics cores for the primal, tangent and adjoint solver.

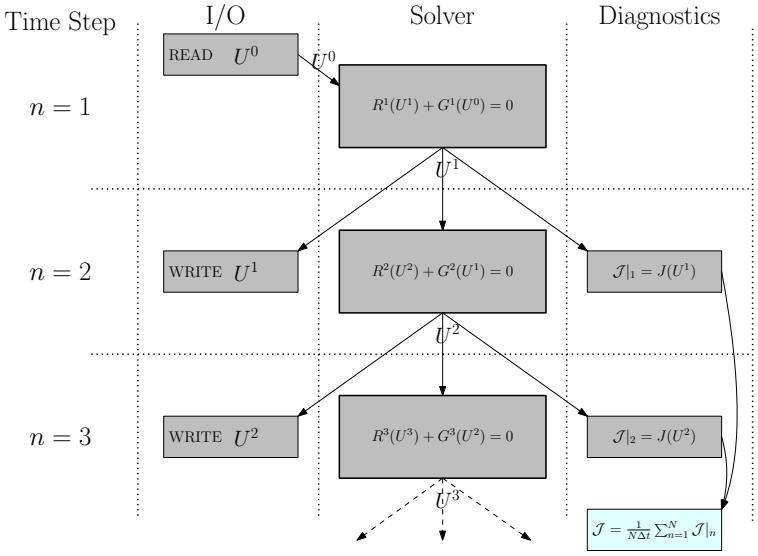


Figure 3. Flow chart for the primal solver for the first three time steps of a run. The notation is the same as that used in Figure 2.

There are two main advantages to using dedicated cores for I/O and diagnostics. Firstly, it provides the flexibility to carry out time-consuming tasks like reading and writing a time-slab to disk while computing the solution at the next time step. Figure 3 illustrates overlapping I/O, the solver and diagnostics for the primal solver. A similar scheme is used for the tangent and adjoint solvers. By overlapping I/O and computation, the tangent and adjoint solvers can compute a time-slab faster than the primal solver.[33] The

second advantage of using asynchronous I/O and diagnostics is that the number of cores set aside for I/O can be made fairly small to avoid running into scaling issues with parallel I/O.

III. Example Physical Models

The modular framework considered in this study is designed to support a wide variety of physical models and discretizations. As sketched in Figure 4, the physical models chosen to solve the particular fluid/structure problem of the capsule and its parachutes are described here, together with the associated finite-element methods.

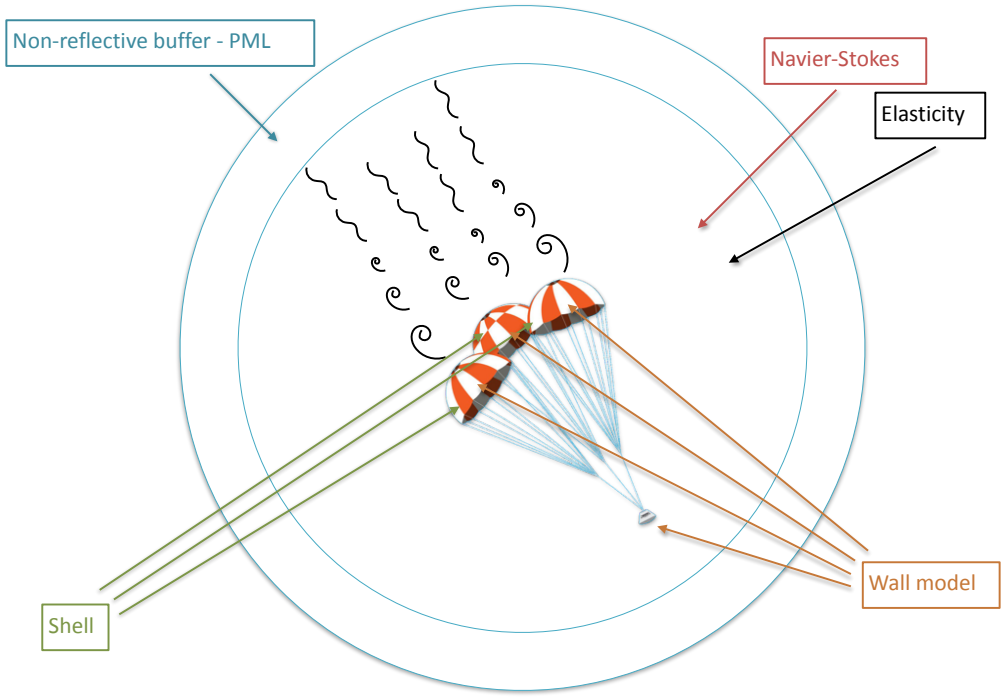


Figure 4. Sketching of the parachute/capsule system and the associated physical models.

The fluid/mesh/structural solvers may be viewed as three coupled physics models which must be solved concurrently in order to perform the FSI simulation. Additional physical models have been developeds to model boundary conditions such as the Perfectly Matched Layer method (PML)[18] or wall models.[20] Future applications of interest may include multi-species reacting flows or surface ablation problems. Each of the physical models may use a different finite-element discretization and take advantage of different solution strategies. Due to the large number of physical models and discretizations which we wish to incorporate in our simulations, we have begun developing a modular solution strategy and software framework allowing for members of our group effectively leverage each of our domain expertise.

III.A. Fluid (Compressible Navier-Stokes Equations)

For the fluid domain we solve the compressible Navier-Stokes equations written in conservative form as

$$\boldsymbol{u}_{,t} + \left(\boldsymbol{F}^{Inv} - \boldsymbol{F}^{Visc}\right)_{,i} = 0, \tag{1}$$

where $\boldsymbol{u} = [\rho, \rho u_j, \rho E]$ is the conservative state vector, ρ is the density of the fluid, u_j is the jth velocity component and E is the total energy. The inviscid and viscous fluxes are given respectively by

$$\boldsymbol{F}^{Inv} = \begin{bmatrix} \rho u_i \\ \rho u_j u_i + p\delta_{ij} \\ \rho H u_i \end{bmatrix} \qquad \boldsymbol{F}^{Visc} = \begin{bmatrix} 0 \\ \tau_{ij} \\ \tau_{ij} u_j + \kappa_T T_{,j} \end{bmatrix} \tag{2}$$

where p is the static pressure, δ_{ij} is the Kronecker delta, H is the total enthalpy, τ_{ij} is the viscous stress tensor, T is the temperature and κ_T is the thermal conductivity. The system is closed using the following relationships

$$T = \frac{p}{\rho R}, \qquad p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho u_k u_k\right), \qquad \tau_{ij} = \mu(u_{i,j} + u_{j,i}) - \lambda u_{k,k}\delta_{ij}, \tag{3}$$

where R is the gas constant, γ is the specific heat ratio, μ is the dynamic viscosity and $\lambda = \frac{2}{3}\mu$ is the bulk viscosity.

We use a space-time discontinuous-Galerkin discretization of (1). We seek a solution $\mathbf{u}$ which satisfies the weak form

$$\sum_{\kappa}\left\{\int_I\int_\kappa -\left(\mathbf{w}_{,t}\mathbf{u} + \mathbf{w}_{,i}(\boldsymbol{f}_i^I - \boldsymbol{f}_i^V)\right) + \int_I\int_{\partial\kappa}\mathbf{w}(\widehat{\boldsymbol{f}_i^I\boldsymbol{n}_i} - \widehat{\boldsymbol{f}_i^V\boldsymbol{n}_i}) + \int_\kappa \mathbf{w}(t_-^{n+1})\mathbf{u}(t_-^{n+1}) - \mathbf{w}(t_+^n)\mathbf{u}(t_-^n)\right\} = 0 \tag{4}$$

where the second and third integrals arise due to the spatial and temporal discontinuity, respectively, of the basis functions. $\widehat{\boldsymbol{f}_i^I\boldsymbol{n}_i}$ and $\widehat{\boldsymbol{f}_i^V\boldsymbol{n}_i}$ denote single valued numerical flux functions approximating, respectively, the inviscid and viscous fluxes at the spatial boundaries of the elements.

III.B. Elasticity

We solve the equations of linear elasticity to obtain the volume displacement of the fluid mesh given the prescribed motion of the surface (or part of the surface) of the fluid mesh. The equations of linear elasticity are

$$\sigma_{ij,j} \quad = \quad 0 \tag{5}$$

where σ_{ij} is the Cauchy stress tensor given by

$$\sigma_{ij} \quad = \quad 2\mu_e\epsilon_{ij} + \lambda_e\epsilon_{kk} \tag{6}$$

$\mu_e = \frac{E}{2(1+\nu)}$ and $\lambda_e = \mu_e\frac{2\nu}{1-2\nu}$ are the Lamé constants given as a function of the Young's Modulus, E, and Poisson ratio, ν. The strain tensor, ϵ is given by

$$\epsilon_{ij} \quad = \quad \tfrac{1}{2}\left(u_{i,j} + u_{j,i}\right) \tag{7}$$

where $\mathbf{u}$ is the displacement field. A compact representation of the stress tensor may be given by $\sigma_{ij} = \mathbf{C}^{ijkl}u_{i,j}$, where $\mathbf{C}^{ijkl}$ is the stiffness tensor. We note that we could include an acceleration term in (5) and solve the equations of linear elasticity as a structural model, though we have yet to apply it for this purpose. When applying the linear elasticity model for mesh deformation, the parameters E and ν may be varied spatially to improve mesh quality. A common choice, employed here, is to fix ν and vary E on each element proportionally with the inverse of the Jacobian of mapping from reference to physical space.[34]

We apply a continuous finite-element discretization of (5) over the initial mesh of the fluid domain. Defining $\mathcal{V} = \left\{\mathbf{w} \in \mathcal{H}_1(\Omega \times I), \mathbf{w}|_\kappa \in [\mathcal{P}(\kappa \times I)]^3\right\}$, the space-time finite-element space consists of $\mathcal{C}^0$ continuous piece-wise polynomial functions on each element.

We seek solutions $\mathbf{u} \in \mathcal{V}_E$ satisfying

$$\sum_{\kappa}\left\{-\int I\int_{\kappa^0}\mathbf{w}_{i,j}\mathbf{C}^{ijkl}\mathbf{u}_{k,l} + \int_I\int_{\partial\kappa^0\cap\partial\Omega}\mathbf{w}_i\widehat{\mathbf{C}^{ijkl}\mathbf{u}_{k,l}}\boldsymbol{n}_j\right\} = 0. \tag{8}$$

We note that the integration is performed over the initial spatial mesh (i.e elements κ^0) of the fluid domain extruded in time, as opposed to the deformed mesh. An alternative approach is to integrate over the initial spatial mesh for each time-slab, which may potentially allow for larger mesh deformations. However, this later approach results in a scheme where the mesh deformation is a function of not only the given boundary displacement but the entire displacement history. More details about the linear elasticity solver can be found in Diosady and Murman.[35]

III.C. Shell

The fabric of parachutes is modeled using the linear shell equations, which are the same as the linear elasticity equations; however, under the assumption of simplified shell kinematics one needs to solve only for the displacement of the mid-plane of the shell. This reduces the dimensionality of the problem by one. The weak form of the shell equations are given by:

$$\sum_{\kappa}\left\{\int_I\int_{\kappa^0}\left(-\mathbf{w}_{i,t}\mathbf{y}_{i,t}+H_{\alpha\beta}(\mathbf{w})\frac{Eh}{1-\nu^2}C^{\alpha\beta\gamma\theta}H_{\gamma\theta}(\mathbf{y})+B_{\alpha\beta}(\mathbf{w})\frac{Eh^3}{1-\nu^2}C^{\alpha\beta\gamma\theta}B_{\gamma\theta}(\mathbf{y})+\mathbf{w}_i\tau_{ij}\mathbf{y}_j\right)\right\}=0 \quad (9)$$

The terms for the above equation are defined in detail in Burgess et al.[36]

As with the elasticity model, we integrate over the initial (reference) surface in the spatial direction. The first term corresponds to an acceleration term, the second and third terms correspond to internal energies due to in-plane and bending strains respectively, while the final term corresponds to work done by external forcing (i.e. forcing due to surface tractions τ_{ij} from the fluid). The in-plane and bending strain terms are functions of the displacement $\mathbf{y}$, (see Burgess *et al.*[36] for full details). We note that we directly discretize the structural velocity $\mathbf{y}_{,t}$ using a basis which is piece-wise discontinuous in time and piece-wise continuous in space. The displacement $\mathbf{y}$ which is $\mathcal{C}^0$ continuous in both space and time, is given by directly integrating $\mathbf{y}_{,t}$.

III.D. Perfectly Matched Layer Method

In order to limit spurious acoustic reflections from boundaries of the fluid domain we use a Perfectly Matched Layer (PML) technique on far field boundaries. The PML technique involves solving the compressible Navier-Stokes equations with a forcing term and a set of auxiliary equations corresponding to each conservation law in a buffer region near the boundary of the fluid domain. Details of the PML technique are given in Garai *et al.*[18] In this multi-physics context, the PML technique may be viewed as solving a different physical model in the buffer region coupled to the fluid region through a Riemann solver at the interface of the fluid and buffer regions. The PML region is discretized using a space-time discontinuous-Galerkin finite-element method in the same manner as the fluid region. For simplicity, we assume a fixed spatial mesh in the PML region.

III.E. Wall Model

Due to the large range in scales present in turbulent flows, the resolution required to fully resolve the near-wall region makes Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES) prohibitively expensive for high-Reynolds number wall-bounded flows. Modeling the near-wall region is necessary in order to make industrial computations practical. In Carton de Wiart and Murman,[20] the coupling of the space-time DG solver with several simple wall modeling approaches have been studied. In these modeling approaches, the inner part of the boundary layer is not computed but information from the fluid away from the wall is used to evaluate a modified boundary condition which is then applied at the wall. In the simplest cases the wall model may consist of analytic equations, for example based on the logarithmic law. More complex models, for instance, based on thin boundary layer approximations or integral wall models, can be used to increase the accuracy of the wall model. We therefore treat the wall model as a separate physics module, instead of a simple boundary condition. By defining simple input/output to the fluid domain, we are able to experiment with different modeling approaches without making any major modifications to the fluids module. The wall model is discretized using an appropriate finite-element method on a surface mesh of the fluid domain.

IV. A Multi-Physics Monolithic Solver

In this section, the strategy to transform the "single-physics" solver into a multi-physics monolithic solver is described. The capsule/parachute system and the accompanying physical model presented in Figure 4 is used here to illustrate the coupling procedure and the associated challenges.

The multi-physics domain is split into "regions", each corresponding to the discretization of a single physics model on a given computational mesh. Each region is responsible for computing local residual terms and auxiliary data, which could be sent to other regions. For the capsule/parachute system described in Figure 4, the elasticity and Navier-Stokes physics are defined as separate regions, although they are defined

on the same mesh partition. The same goes for the shell and the wall model modules on the parachutes. They are also defined as separated regions, even though they are defined on the same boundary mesh. Splitting the global problem based on physics/mesh/discretization allows the solver to load balance each region independently. It also gives a lot of flexibility as new physics or discretizations can be added into the solver without impacting the other modules.

We can define the global problem as multiple regions, linked together through coupling operations. As stated in Section II, each region can also be split into three sub-regions, each responsible for specific work: solver, I/O and diagnostics. This is handled in the framework by defining groups of cores, where group is associated with one region and one set of operations (solver, I/O or diagnostics), creating a matrix of communicators, linked together by intra- and inter-communicators. Inter-communicators are used by the coupling objects to communicate data from one region to another. They are also used within a region to send the solution from the solver nodes to the I/O or diagnostics nodes, or the other way around. Intra-communicators encompass a group of (sub-)regions and are used mostly to synchronize the regions together, for instance by broadcasting the value of the time when restarting a simulation by reading an initial solution or gathering the residual norm at the non-linear/linear solver levels. Figure 5 illustrates the matrix of communicators in the context of the capsule/parachute system. Communicators are only created between

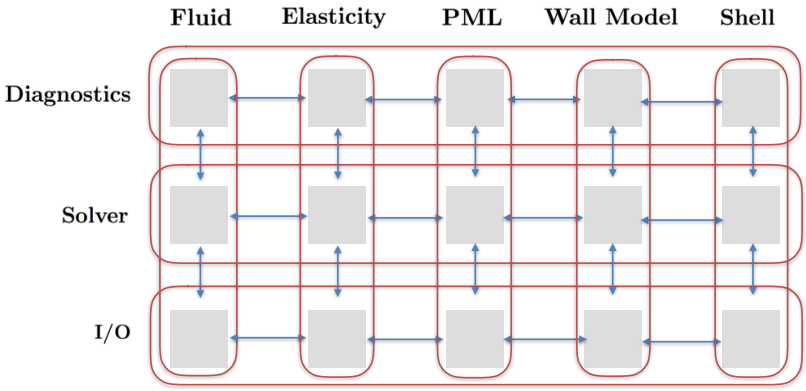


Figure 5. Matrix of communicators between the different sub-regions. Local communicators and denoted by gray squares, intra-communicators by red grouping and inter-communicators by blue arrows.

rows and columns, not diagonally. As regions can also be locally partitioned, local communicators are used to communicate the solution between the different region partitions.

Even though the global problem is split into several regions, it is still tightly coupled by communicating the appropriate data between the different regions (*i.e.* boundary state, boundary stress, displacement, etc.). These coupling data are then used inside boundary conditions or domain forcing functions. As stated in Section II, the framework uses a space-time finite-element method, which requires the solution of a non-linear system at each time step using a Newton-Krylov approach. This gives three main loops in the system: a temporal loop, a non-linear solver loop and a linear solver loop. The different loops are presented in Figure 6, showing where the coupling needs to occur. The coupling between the different regions will occur at every level of the code, from the temporal loop all the way down to the linear solver. In the temporal loop, coupling may occur after solving the non-linear system in order to compute diagnostics that can depend on coupling data (cut planes, average, boundary integrals, etc.). The coupling also occurs at every non-linear and linear solver iteration. Indeed, in a monolithic approach, the state $\mathbf{u}$ and residual $\mathbf{R}$ vectors are formed over all degrees of freedom corresponding to the coupled equations, therefore spanning all the different regions (*i.e.* fluid/mesh/structure/PML/wall-model)

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_n \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_n \end{bmatrix}. \tag{10}$$

The residual vector can be split into multiple residual vectors $\mathbf{R}_i$, each vector associated with one region

$$\mathbf{R}_i \quad = \quad \mathbf{R}_i(\mathbf{u}_i, \boldsymbol{\alpha}_{i1}, \boldsymbol{\alpha}_{i2}, \ldots, \boldsymbol{\alpha}_{ij}), \tag{11}$$

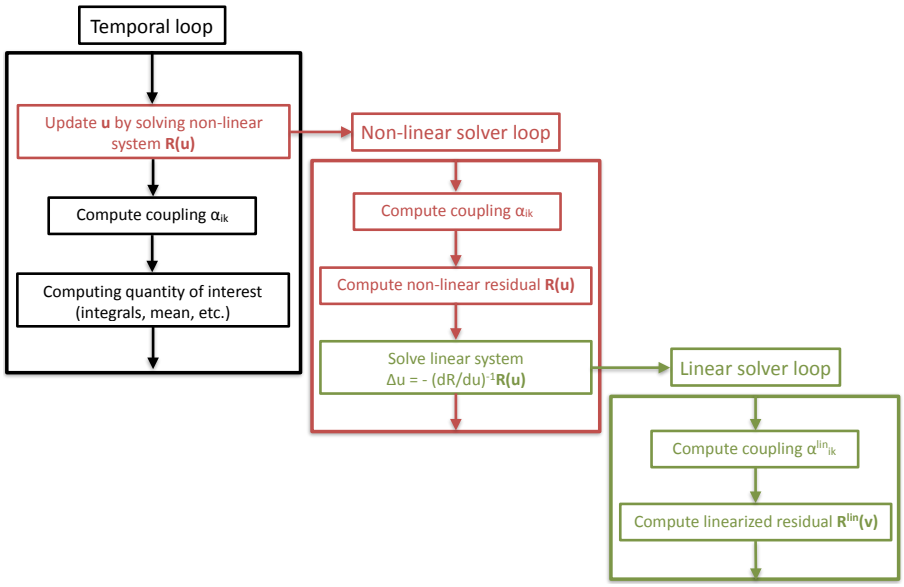


Figure 6. Temporal (black), non-linear solver (red) and linear solver (green) loops with associated operations.

where $\mathbf{u}_i$ the local vector state and $\boldsymbol{\alpha}_{ij} = \boldsymbol{\alpha}_{ij}(\mathbf{u}_j, \boldsymbol{\alpha}_{jk})$ represents the vector of coupling data shared between regions i and j (which can also depend on the coupling data $\boldsymbol{\alpha}_{jk}$ between region j and another region k). At each time-slab, the non-linear system

$$\mathbf{R}(\mathbf{u}) = 0 \tag{12}$$

must be solved globally. The same Jacobian-free Newton-Krylov scheme previously developed for the fluid module[14] is used to drive the residual towards zero. The Newton-Krylov solver involves the repeated evaluation of residuals and linearized residuals at each Newton and Krylov iteration, respectively. The Newton solver updates the state at each iteration using

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \Delta\mathbf{u}^k \quad \text{with} \quad \Delta\mathbf{u}^k = -\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right)^{-1} \mathbf{R}(\mathbf{u}^k), \tag{13}$$

where k is the Newton iteration number. This linear system is then solved using a Krylov method in which linearized residual vectors $\mathbf{R}_i^{lin}(\mathbf{u}_i, \boldsymbol{\alpha}_{ij}; \mathbf{u}_i^{lin}, \boldsymbol{\alpha}_{ij}^{lin})$ are evaluated, where $\mathbf{u}_i$ and $\boldsymbol{\alpha}_{ij}^{lin}$ are the linearized state and linearized coupling data, respectively. The evaluation of the linearized residual involves similar interactions between modules.

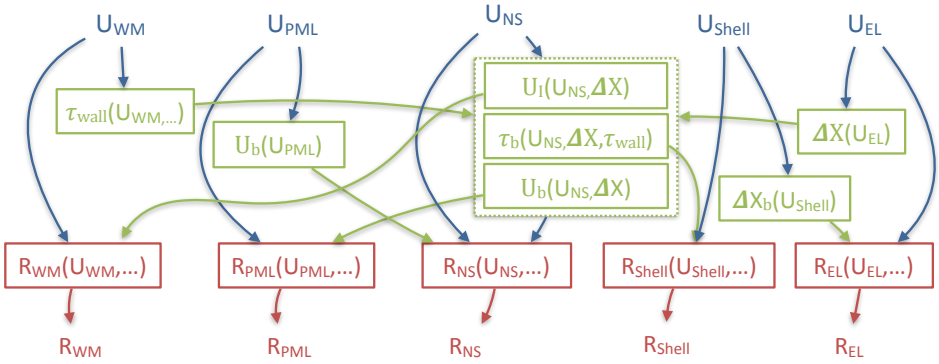


Figure 7. Coupling of physics modules for primal residual evaluation. U denotes the state vector, while R denotes the residual vector. Each box depict a physics module; WM: wall model, PML: perfectly matched layer buffer region, NS: Navier-Stokes equations for fluid region, Shell: linear shell model, EL: linear-elastic mesh deformation. The quantities transferred between different physics modules are wall stress, τ_w, boundary state U_b, extracted state for fluid volume U_e, boundary displacement X_w, mesh coordinates X.

The coupling operation can require a very complex communication pattern, depending on the problem. Figure 7 depicts the coupling of the different physics modules required to compute the non-linear residual of the capsule/parachute system described in Figure 4. The complexity of the coupling is clearly seen in Figure 7. Some couplings depend on other couplings (for instance the traction sent from Navier-Stokes to shell depends on the wall friction given by the wall model). As described in Section II, the whole coupling process needs to be reversed for the adjoint solver. The resulting adjoint graph presented in Figure 8 shows that the dependency graph in Figure 7 cannot be reused for the adjoint, adding another layer of complexity into the framework.

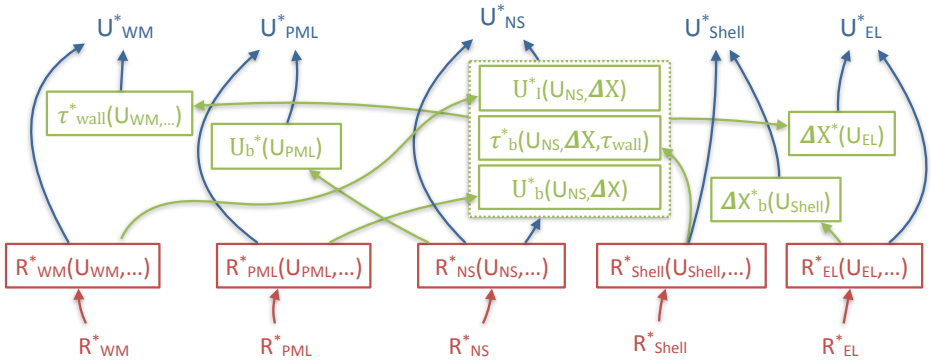


Figure 8. Coupling of physics modules for adjoint residual evaluation. The adjoint of the variables defined in Figure 7 are denoted by $\cdot^*$.

While each physics/discretization model implements the relevant terms in order to compute a residual, some operations in the residual evaluation are inherently sequential, reducing the potential for parallel scalability of the combined scheme. The situation is further complicated when adjoint problems are solved, for which the order of operations is essentially reversed, as shown in Figure 8. Exposing greater granularity within each physics/discretization modules to the multi-physics solver and then using an appropriate graph-based technique to dynamically determine the order of operations is key to reducing some of these sequential bottlenecks. This will be addressed in an upcoming study.

V. Example Applications

We present some sample applications of using the multi-physics formulation described above. The cases serve as examples of problems which may be solved using our multi-physics solver, though each of the following examples represent simplified preliminary calculations.

V.A. Heaving Pitching Airfoil

The first test case we consider is the flow over a heaving pitching airfoil. In particular we consider one of the test cases from the AIAA higher-order workshop.[37] The test case serves as a validation for the moving domain space-time discontinuous-Galerkin fluid solver and for the elasticity based mesh deformation technique. The test case involves flow over the NACA0012 airfoil at $M = 0.2$, $Re = 5000$ initially at zero angle of attack with a prescribed heaving and pitching motion. Figure 9 shows the motion of the airfoil and the computed vorticity in the flow solved using an 8th order solution in both space and time. At this point we have only performed qualitative comparisons to results from the higher-order workshop.

Figure 10 depicts the initial 8th order mesh, along with the mesh deformed using elasticity part-way through the heaving motion. Displacing the interior nodes of the mesh the linear elastic model, with a Poisson ratio of $\nu = 0.4$ and Young's modulus scaled with the inverse of the Jacobian determinant, ensure valid elements everywhere in the domain despite the large deformations due to the prescribed motion.

V.B. DNS of T106 Low Pressure Turbine

Figure 11 presents the instantaneous flow solution from a direct numerical simulation of flow over a low pressure turbine.[18] The simulation was performed using the 8th-order space-time DG formulation using a compressible Navier-Stokes/PML coupling. Using the coupled solution strategy eliminates spurious acoustic

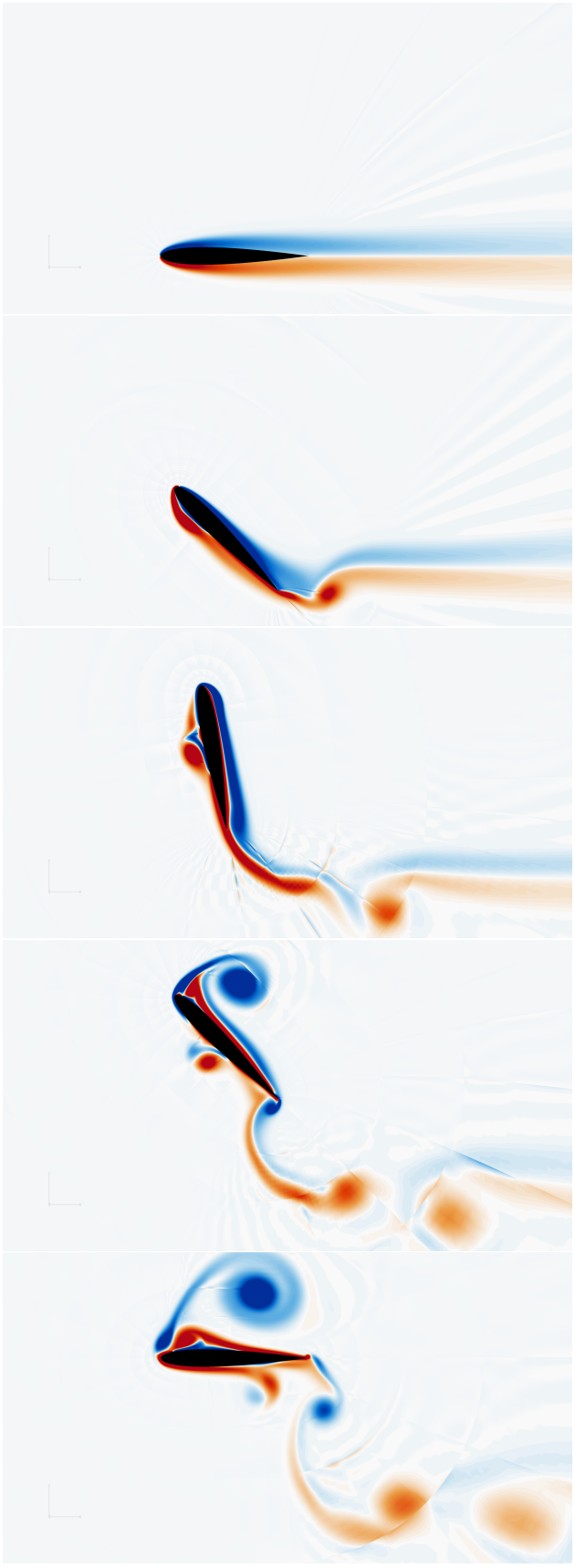


Figure 9. Vorticity magnitude for Heaving/Pitching NACA0012 airfoil.

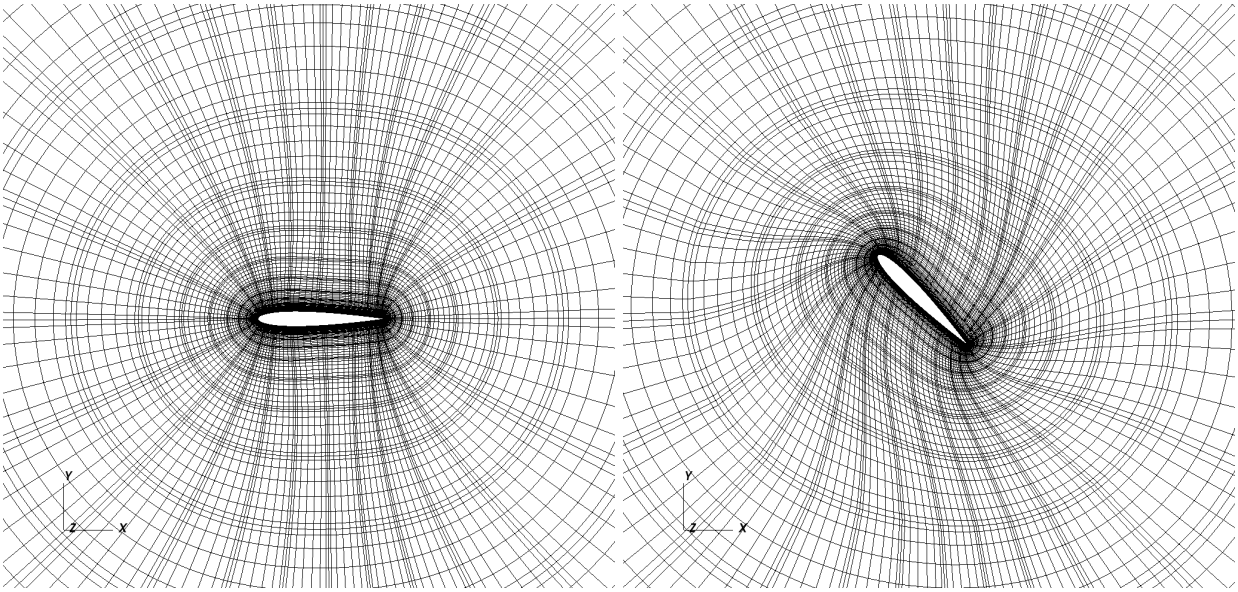


Figure 10. Initial and displaced mesh using linear elasticity for a heaving/pitching NACA0012 airfoil.

reflections from the far-field boundary seen using a single model Navier-Stokes solve using Riemann-based boundary conditions. The corresponding density gradient is also shown in Figure 11 from which the improved quality of the coupled simulation is clearly evident.

V.C. NACA4412 with wall model

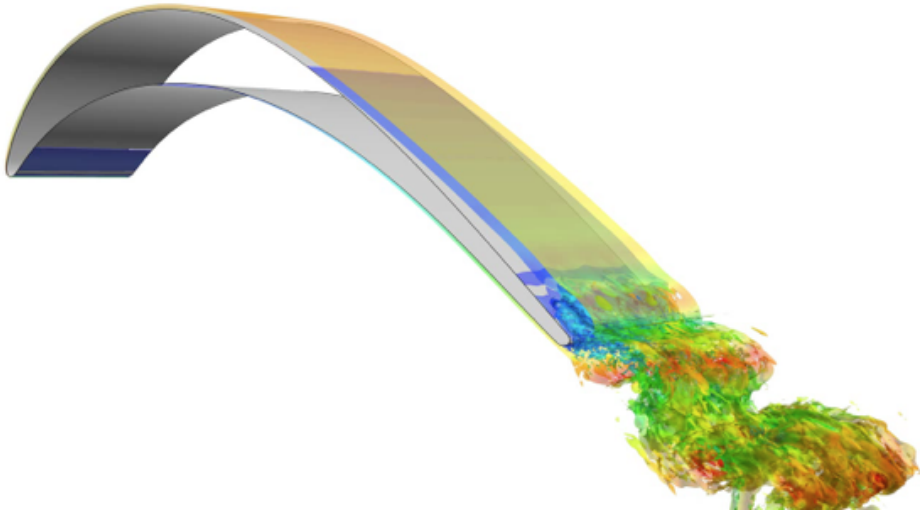
Figure 12 shows the instanteneous iso-contours of vorticity for a simulation of separated flow over a near stall NACA4412 airfoil at $Re = 1.6 \times 10^6$ using the compressible Navier-Stokes fluid solver coupled with an algebraic equilibrium wall model.[20] The simulation is performed on a 2D-unstructured mesh extruded in the spanwise direction leading to a total number of spatial degrees of freedom equal to $5.12M$. The wall model allows one to drastically reduce the computational cost of wall-bounded high Reynolds number flows by modeling the effects of thin turbulent boundary layers. More complex models will be investigated in the near future.

V.D. Dynamic simulation of an Apollo parachute

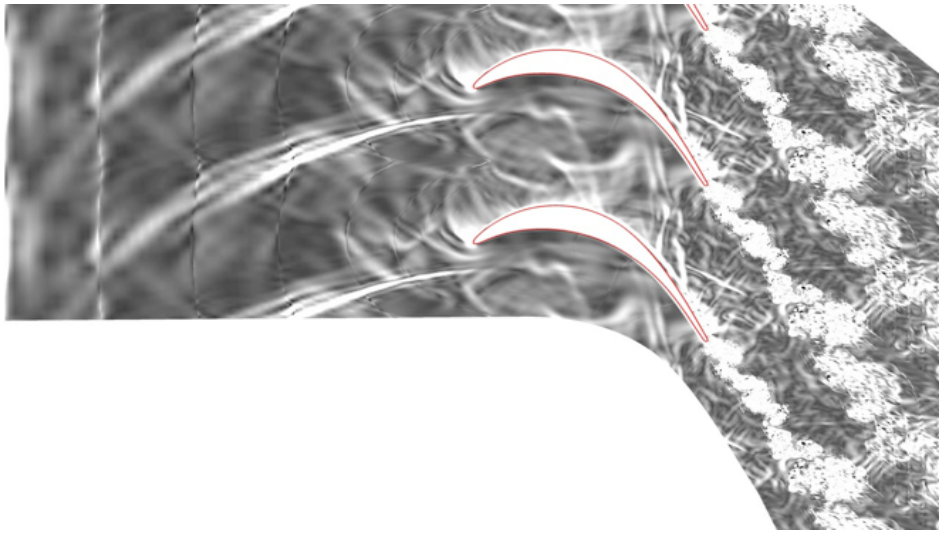
Figure 13 depicts a moving domain simulation of flow about an oscillating parachute. This preliminary simulation was performed on an unstructured mesh with approximately $16M$ spatial degrees of freedom using the space-time discontinuous-Galerkin discretization with second order elements. The motion of the mesh is prescribed and simple enough such that the mesh displacement is given by an analytic function.

VI. Towards a graph-based solver

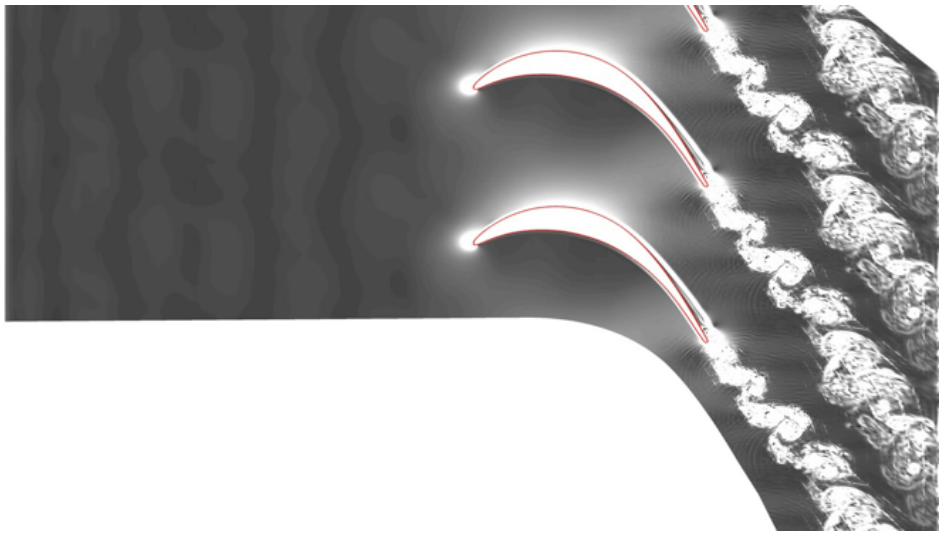
The computational cost of the monolithic solver is mainly driven by the computation of the region-local residual, the computation of the coupling objects, and the communications between the different regions. As seen in Sections II and IV, the residual and the coupling involve many communications of data between regions or region partitions. As we aim for very large problems running on thousands of cores, we need be sure that the communications do not become the dominant part of the computational time. Non-blocking communications overlapping with effective computational work solves this issue, allowing communication of large data without significantly impacting the total computational time. This assumes that the effective computational time is well balanced with the communication time. For the moment, the dependency between the effective computational time and the communication time is hardcoded in the process, both for the computation of the residual and the coupling objects. The hardcoded dependency graph is also separated between the residual and the coupling objects, meaning that the residual operations cannot be used to hide



(a) Isosurfaces of Vorticity.



(b) Density Gradient using Riemann BC.



(c) Density Gradient using PML.

Figure 11. Direct numerical simulation of a T106 low pressure turbine with and without PML.[18]

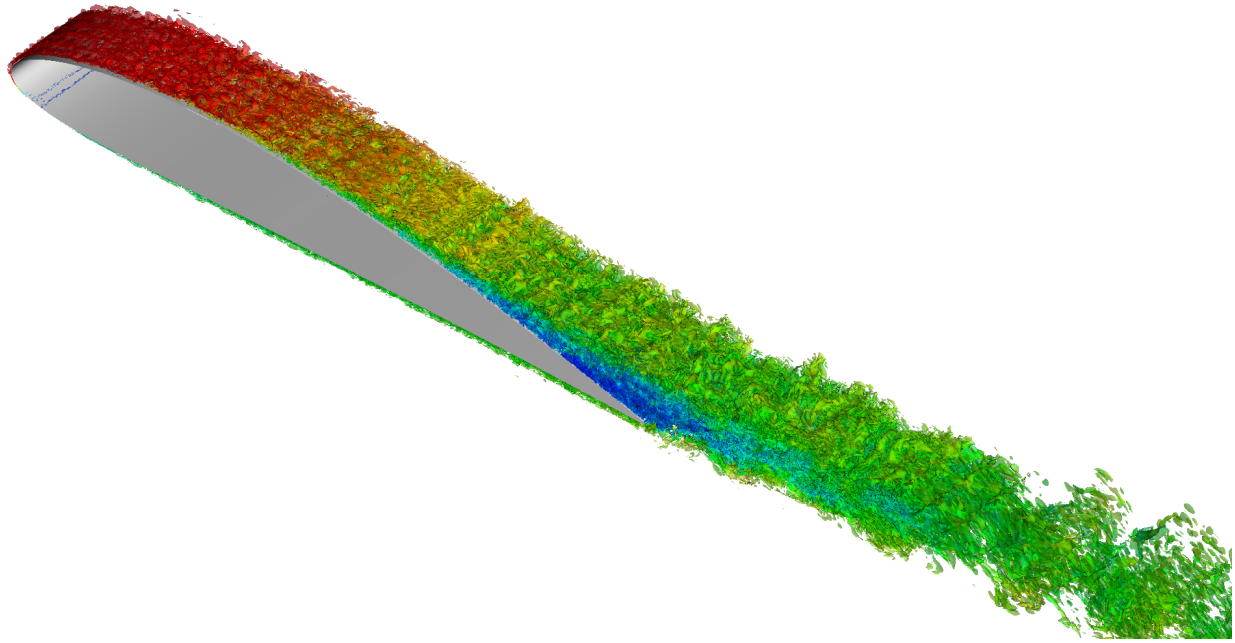


Figure 12. NACA4412 at near-stall conditions with instantaneous iso-contours of vorticity colored by velocity magnitude.[20]

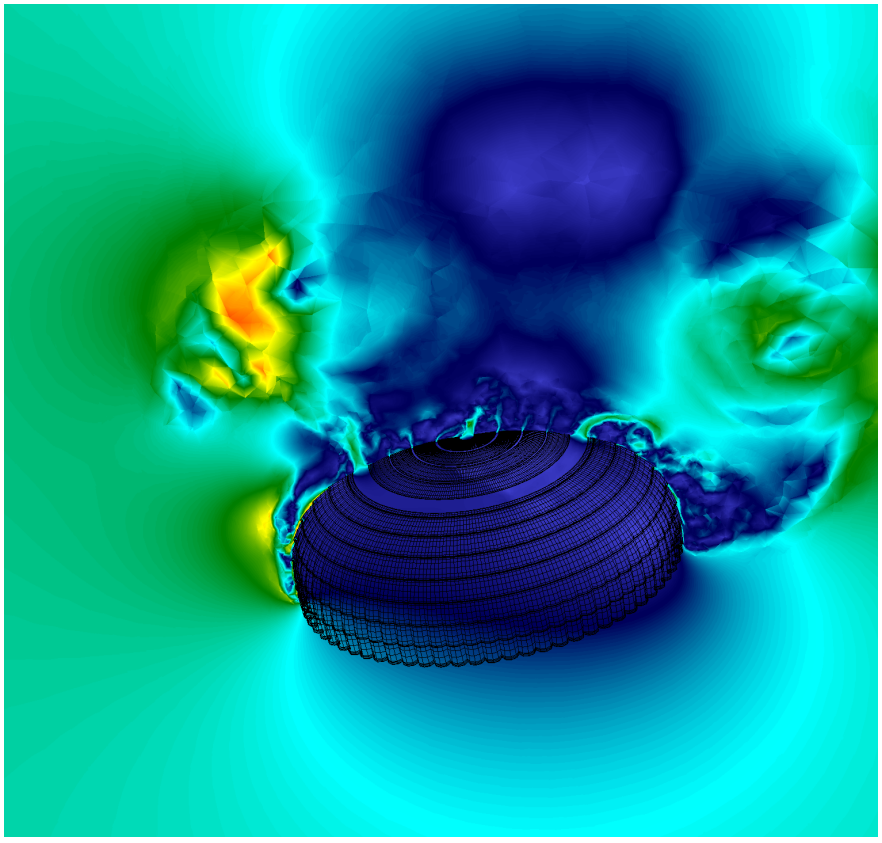
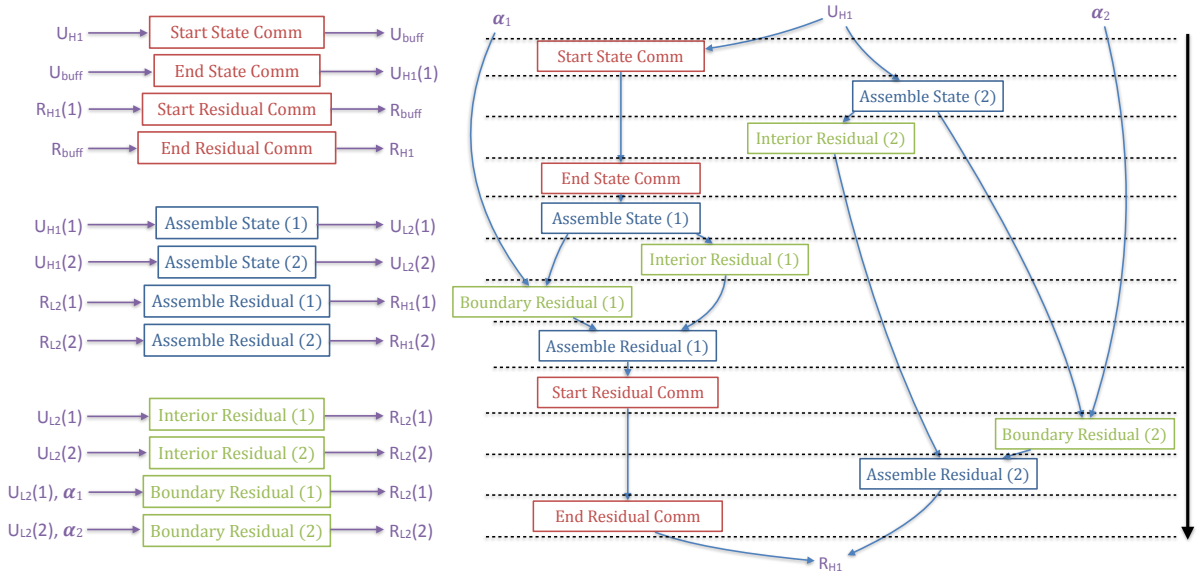


Figure 13. Mach number contours of a dynamic simulation of an Apollo parachute.

communications for the coupling. As shown in Figure 7, very complex multi-physics problems can have coupling objects with convoluted dependencies on one another. In order to effectively hide communication and efficiently load balance the computation, the execution order must be specified for each multi-physics problem.

Manually specifying the execution order for each problem is time-consuming and potentially error prone, thus an approach based on directed-acyclic graphs (DAG)[38,39] is considered in order to automate the workflow. Each operation may be viewed as a node of a graph, while the directed edges denote data-dependencies between the different operations. In the proposed approach, a list of required operations and their associated dependencies are specified, which is then used to automatically construct the operation graph. The correct order of operations is determined using a graph traversal algorithm. A very complicated set of operations can thus be assembled from smaller individual components. Figure 14 presents an example of using the approach for evaluating the CG residual. Figure 14 also shows a list of required operations (communications, finite-element assembly, elemental residual evaluations) and their associated data dependencies. Using these dependencies we can construct the operation graph. Figure 14 shows the associated graph, which can be traversed in several different orders. In particular, the order of execution can be dynamically adjusted in order to appropriately mask the communications. The same procedure can be used to generate the execution graph for the DG and C^1-DG residuals. Small prototypes have already been successfully implemented, showing an increase in the modular flexibility of the framework. We continue to extend this approach to the coupling terms allowing interleaving of coupling and residual operations to improve load balancing. We intend to further explore this approach in upcoming studies.



(a) List of Operations

(b) Operator Graph and Traversal 1

Figure 14. DAG approach to CG residual evaluation.

VII. Conclusion

This work presented the challenges associated with the design of a modular multi-physics high-order space-time finite-element framework. Starting from a single-physics, single-discretization solver, the framework has been extended to allow a monolithic coupling of different physics modules with different finite-element discretizations. Building an appropriate matrix of communicators and defining coupling objects between the different physics modules allowed us to reach this goal. The physics modules are coupled at every level of the computation, from the temporal loop all the way down to the linear solver. Multiple physics modules have been implemented with the objective of simulating the dynamics of a capsule/parachute system. Examples of couplings between these physics modules have been presented, showing the maturity of the framework. The coupling between Navier-Stokes and the Perfectly Matched Layer method have been presented on the

challenging scale-resolving simulation of a low pressure turbine blade, showing the advantages of the non-reflective approach for the domain inflow and outflow. The ability of the solver to deform a mesh using the linear elasticity equation and exchange the updated mesh with Navier-Stokes has been shown for a heaving/pitching airfoil. Wall modeling has also been tested in order to decrease the cost of running wall-bounded high-Reynolds number flows. Treating the wall model as a separate physics enables better load balancing of the problem and greater flexibility as the different models can be easily tested in the same framework. Finally, preliminary computation of a single parachute with no capsule have been performed, paving the way to a fully coupled simulation of the capsule/parachute system.

Many improvements can be made to the framework. The efficieny and robustness of the non-linear solver needs to be further improved and more advanced preconditioning techniques need be implemented. The impact of coupling multiple physics on the non-linear system need to be further analyzed and the different physics modules need to be tuned in order to obtain a better conditioned system. For the moment, the load balancing of the global multi-physics system is done by the user. A more automated approach will be considered, where the software will automatically distribute the cores between the different regions as a function of the size of the problem and the cost of each physics/discretization. Finally, a graph-based approach will be considered in order to remove the need of hard-coding the dependency graph between the different coupling/residual operators. This will allow a better load balancing of the operations whilst improving the global modularity of the framework.

References

[1]Keyes, D., McInnes, L. C., and CarolWoodward, "Multiphysics Simulations: Challenges and Opportunities," Argonne National Laboratory ANL/MCS-TM-321, 2011.

[2]Ray, E. S., "Test Vehicle Forebody Wake Effects on CPAS Parachutes," AIAA Paper AIAA-xxxx, 2017.

[3]Heil, M., Hazel, A. L., and Boyle, J., "Solvers for large-displacement fluid–structure interaction problems: segregated versus monolithic approaches," *Computational Mechanics*, Vol. 43, 2008, pp. 91–101.

[4]McDaniel, D. R., Tuckey, T., and Morton., S. A., "The HPCMP CREATETM-AV Kestrel Computational Environment and its Relation to NASA's CFD Vision 2030," AIAA 2017-0813, 2017.

[5]McDaniel, D. R. and Tuckey, T., "Multiple Bodies, Motion, and Mash-Ups: Handling Complex Use-Cases with Kestrel," AIAA 2014-0415, 2014.

[6]Jayaraman, B., Wissink, A., Shende, S., Adamec, S., and Sankaran, V., "Extensible Software Engineering Practices for the Helios High-Fidelity Rotary-Wing Simulation Code," AIAA 2011-1178, 2011.

[7]Wissink, A. M., Sitaraman, J., Sankaran, V., Mavriplis, D. J., and Pulliam, T. H., "A Multi-Code Python-Based Infrastucture for Overset CFD with Adaptive Cartesian Grids," AIAA 2008-0927, 2008.

[8]Atwood, C. A., Adamec, S. A., Murphy, M. D., Post, D. E., and Blair, L., "Collaborative software development of scalable DoD computational engieering," *DoD HPCMP User Group Conference*, 2010.

[9]Kiris, C. C., Barad, M. F., Housman, J. A., Sozer, E., Brehm, C., and Moini-Yekta, S., "The LAVA Computational Fluid Dynamics Solver," AIAA 2014-0070, 2014.

[10]Gaston, D., Newman, C., Hansen, G., and Lebrun-Grandié, D., "MOOSE: A parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering and Design*, Vol. 239, 2009, pp. 1768–1778.

[11]Farhat, C., van der Zee, G., and Geuzaine, P., "Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear compuational aeroelasticity," *CMAME*, Vol. 195, 2006, pp. 1973–2001.

[12]van Brummelen, E., Hulshoff, S., and de Borst, R., "A monolithic approach to fluid-structure interaction using space-time finite elements," *Comput. Methods Appl. Mech. Engrg.*, Vol. 2003, 2003, pp. 2727–2748.

[13]Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., "Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Adjoint Derivative Computations," *AIAA Journal*, Vol. 52, 2014, pp. 935–951.

[14]Diosady, L. T. and Murman, S. M., "Design of a variational multiscale method for turbulent compressible flows," AIAA Paper 2013-2870, 2013.

[15]Diosady, L. T. and Murman, S. M., "DNS of flows over periodic hills using a discontinuous Galerkin spectral element method," AIAA Paper 2014-2784, 2014.

[16]Diosady, L. T. and Murman, S. M., "Tensor-Product Preconditioners for Higher-order Space-Time Discontinuous Galerkin Methods," *Journal of Computational Physics*, Vol. 330, 2017, pp. 296–318.

[17]Diosady, L. T. and Murman, S. M., "Higher-Order Methods for Compressible Turbulent Flows Using Entropy Variables," AIAA Paper 2015-0294, 2015.

[18]Garai, A., Diosady, L., Murman, S., and Madavan, N., "Development of a Perfectly Matched Layer Technique for a Discontinuous-Galerkin Spectral-Element Method," AIAA 2016-1338, 2016.

[19]Ceze, M., Diosady, L. T., and Murman, S. M., "Development of a High-Order Space-Time Matrix-Free Adjoint Solver," AIAA Paper 2016-0833, 2016.

[20]Carton de Wiart, C. and Murman, S. M., "Assessment of Wall-modeled LES Strategies Within a Discontinuous-Galerkin Spectral-element Framework," AIAA 2017-1223, 2017.

[21]Blonigan, P., Fernandez, P., Murman, S., Wang, Q., G., R., and Magri, L., "Toward a chaotic adjoint for LES," *Center for Turbulence Research Proceedings of the Summer Program 2016*, 2016.

[22]Orszag, S. and Patterson, G.S., J., "Numerical simulation of turbulence," *Statistical Models and Turbulence*, edited by M. Rosenblatt and C. Atta, Vol. 12 of *Lecture Notes in Physics*, Springer Berlin Heidelberg, 1972, pp. 127–147.

[23]Morinishi, Y., Lund, T. S., Vasilyev, O. V., and Moin, P., "Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow," *Journal of Computational Physics*, Vol. 143, 1998, pp. 90–124.

[24]Honein, A. E. and Moin, P., "Higher entropy conservation and numerical stability of compressible turbulence simulations," *Journal of Computational Physics*, Vol. 201, 2004, pp. 531–545.

[25]Subbareddy, P. K. and Candler, G. V., "A fully discrete, kinetic energy consistent finite-volume scheme for compressible flows," *Journal of Computational Physics*, Vol. 228, No. 5, 2009, pp. 1347 – 1364.

[26]Sjögreen, B. and Yee, H., "On Skew-Symmetric Splitting and Entropy Conservation Schemes for the Euler Equations," *Numerical Mathematics and Advanced Applications 2009*, edited by G. Kreiss, P. Lötstedt, A. Målqvist, and M. Neytcheva, Springer Berlin Heidelberg, 2010, pp. 817–827.

[27]Lesoinne, M. and Farhat, C., "Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations," *Comput. Methods Appl. Mech. Engrg.*, Vol. 134, 1996, pp. 71–90.

[28]Hubner, B., Walhorn, E., and Dinkler, D., "A monolithic approach to fluid-structure interaction using space-time finite elements," *Comput. Methods Appl. Mech. Engrg.*, Vol. 2004, 2004, pp. 2087–2104.

[29]Tezduyar, T. E. and Sathe, S., "Modelling of fluid-structure interactions with the space-time finite elements: Solution techniques," *Int. J. Numer. Meth. Fluids*, Vol. 54, 2007, pp. 855–900.

[30]Wang, L. and Persson, P.-O., "A high-order discontinuous Galerkin method with unstructured space-time meshes for two-dimensional compressible flows on domains with large deformations," *Computers and Fluids*, Vol. 118, 2015, pp. 53–68.

[31]Kirby, R. E., Yosibach, Z., and Karniadakis, G. E., "Towards stable coupling methods for high-order discretizations of fluid-structure interaction: Algorithms and observations," *Journal of Computational Physics*, Vol. 223, 2007, pp. 489–518.

[32]Garai, A., Diosady, L., Murman, S., and Madavan, N., "Scale-Resolving Simulations of Bypass Transition in a High-Pressure Turbine Cascade Using a Spectral-Element Discontinuous Galerkin Method," *ASME. J. Turbomach*, 2017.

[33]Blonigan, P. J., "Adjoint sensitivity analysis of chaotic dynamical systems with non-intrusive least squares shadowing," *Journal of Computational Physics*, Vol. 348, 2017, pp. 803–826.

[34]K. Stein, T. T. and Benney, R., "Mesh Moving Techniques for Fluid-Structure Interactions With Large Displacements," *J. Appl. Mech*, Vol. 70, No. 1, 2003, pp. 58–63.

[35]Diosady, L. T. and Murman, S. M., "A linear-elasticity solver for higher-order space-time mesh deformation," *AIAA SciTech Forum, January 2018, Kissimmee, Florida*, 2018.

[36]Burgess, N., Diosady, L., and Murman, S., "A C1-discontinuous-Galerkin Spectral-element Shell Structural Solver," AIAA 2017-3727, 2017.

[37]Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-Order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, 2013, pp. 811–845.

[38]Mikida, E., Jain, N., Gonsiorowski, E., Barnes, Jr., P. D., Jefferson, D., Carothers, C. D., and Kale, L. V., "Towards PDES in a Message-Driven Paradigm: A Preliminary Case Study Using Charm++," *ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (PADS)*, SIGSIM PADS '16 (to appear), ACM, May 2016.

[39]Berzins, M., Meng, Q., Schmidt, J., and Sutherland, J. C., "DAG-Based Software Frameworks for PDEs," *Proceedings of the 2011 International Conference on Parallel Processing*, Euro-Par'11, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 324–333.