Structures, Structural Dynamics, and Materials and Co-located Conferences
April 8-11, 2013, Boston, Massachusetts
54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference

Large-Scale MDO of a Small Satellite using a Novel Framework for the Solution of Coupled Systems and their Derivatives

John T. Hwang [*] , Dae Young Lee [†] , James W. Cutler [‡] , and Joaquim R. R. A. Martins [§]
University of Michigan, Ann Arbor, Michigan, 48109, United States

Gradient-based multidisciplinary optimization is applied to a small satellite design problem. A novel MDO framework is developed by adopting a non-conventional definition of components which yields two key benefits for solving multidisciplinary analysis and optimization problems. The first is a unified solver which generalizes many existing methods for solving nonlinear and linear systems, and the second is an automated and efficient method for computing coupled derivatives. This framework is used to solve a small satellite optimization problem involving several disciplines including orbit dynamics, attitude dynamics, attitude control, temperature, solar power, battery, and communication. Multi-point optimizations involving over 35,000 design variables and 1.2 million unknowns require on the order of 5 hours to converge to significantly improved designs, and this efficient design optimization capability is used to compare potential launch options.

I. Introduction

Satellites serve a multitude of purposes ranging from navigation and scientific research to military applications. Over the past decade, small satellites have gained increasing interest as alternatives to larger satellites because of the low time and cost required to manufacture and launch them. The CubeSat class of small satellites conforms to a set of specifications that facilitates relatively frequent launches as secondary payloads, and as a result, they are becoming a common platform for education and research.

The CubeSat investigating Atmospheric Density Response to Extreme driving (CADRE) mission is an effort funded by the NSF to study the response of the Earth's upper atmosphere to auroral energy inputs [1]. This project addresses the need for more accurate modeling of space weather effects, motivated in part by the growth of the global space-based infrastructure. To help answer some of the important scientific questions in this area, CADRE will provide critical in-situ measurements in the ionospheric and thermospheric regions.

The CADRE CubeSat is to inherit much of the design of the University of Michigan's Radio Aurora eXplorer (RAX) CubeSat; however, the unique scientific goals of the mission necessitate a detailed design study. Power is a driving factor as the scientific instruments are to run continuously for large parts of the mission. To ensure sufficient power can be generated and stored, variables such as battery sizing, solar panel sizing, and attitude must be considered. The tradeoffs involved with these variables draw in a large number of disciplines, resulting in a highly coupled and complex problem not well-suited to design using experience and human intuition.

There are many studies in the literature that propose systematic methodologies for solving such a complex multidisciplinary problem. For instance, Ekpo and George [2] presented a system-based methodology for designing highly adaptive small satellites, while Cvetkovic and Robertson [3] proposed designing small satellites for remote sensing by adapting an existing small satellite design and decomposing the problem into subsystems. Other authors also proposed separating by subsystems, but added the idea of simulation-based design within an integrated framework [4] that also supports hardware-in-the-loop simulations [5] and knowledge-based design and optimization [6]. Spangelo and Cutler [7] performed optimization of both the design of the vehicle and operations, using a Monte Carlo algorithm to search the design space.

The current work seeks to consider the multidisciplinary problem with a greater level of detail and coupling than existing studies. The approach is to focus on the scientific computing side of the problem and use efficient numerical algorithms in an effort to incorporate as much fidelity as possible within a monolithic computational design framework. The motivation comes from the fact that there are many variables over which the designer has control, and it would be desirable to simultaneously optimize the design and operation of CADRE.

[*] Ph.D. Candidate, Department of Aerospace Engineering, AIAA Student Member
[†] Ph.D. Candidate, Department of Aerospace Engineering, AIAA Student Member
[‡] Assistant Professor, Department of Aerospace Engineering, AIAA Member
[§] Associate Professor, Department of Aerospace Engineering, AIAA Associate Fellow

However, there are a number of obstacles that make this a challenging objective. First, there are a large number of disciplines that must be considered. Keeping track of dependencies between disciplines is necessarily a manual task, and these dependencies affect the optimal sequence of execution of codes. It follows that there are a large number of intermediate variables, or variables that represent the state of the system. The relevant disciplines for CADRE include orbit dynamics, attitude dynamics, attitude control, kinematics, thermal, electrical, battery, and communication. Many of the aforementioned disciplines involve ordinary differential equations (ODE) that must be solved numerically with sufficient resolution, resulting in discretizations involving thousands or more degrees of freedom for each quantity. These necessitate sparse linear algebra and efficient solution techniques to meet requirements on memory usage and computation time. Furthermore, we wish to optimize profile variables such as the spacecraft attitude profile over a period of time. Giving an optimizer control over discretized profile variables yields potentially hundreds or thousands of design variables. The design of CADRE also involves multiple time scales as the period of an orbit is on the order of an hour, while the precession of the orbit and the relative position of the sun changes on the order of months, and these affect the power generation capability of the solar panels.

The approach selected is largely driven by these challenges and incorporates many existing techniques from the multidisciplinary design optimization (MDO) field. Due to the large number of optimization variables, we use gradient-based optimization, as gradient-free methods do not perform well for problems with more than hundreds of design variables [8]. Sequential quadratic programming (SQP) methods are relatively efficient for nonlinear optimization problems with equality and inequality constraints. The coupling between disciplines will be resolved using the multidisciplinary feasible architecture (MDF) [9] to ensure that local optima found represent those of the true multidisciplinary problem, capturing the cascading effects that design variables can have across multiple disciplines. When combined with analytic derivative computation [10], gradient-based optimization is a powerful tool as the adjoint method makes it possible to compute derivatives at a cost nearly independent of the number of design variables, while the number of iterations required to converge the optimization scales only linearly. The large number of disciplines and the need to automate the handling of dependencies between disciplines necessitate a novel computational design framework, which will integrate the analytic derivative computation using the MDF architecture.

The structure of this paper is as follows. First, the computational design framework will be presented with subsections on the theory, algorithms, and software design. This will be followed by a detailed description of the numerical tools and methods used to aid in modeling the CADRE CubeSat. Next, each discipline will be discussed in depth, and finally, the formulation and results of the MDO problem will be presented.

II. Multidisciplinary Optimization Framework

There is a significant investment of time and cost associated with the development of computational tools for multidisciplinary analysis and optimization. However, much of this work is independent of the specific design problem and implementation and is a common set of tools useful for any MDO problem. There are many existing computational design frameworks that provide these sets of tools. These frameworks offer some or all of: built-in tools such as optimizers, graphical user interfaces, visualization capabilities, and automatic data transfer.

The framework to be presented in this section differs from these types of frameworks in three ways. First, it is designed for a specific niche, the subset of MDO problems that are to be solved using gradient-based methods. Thus, the key assumption is that all functions in the framework are differentiable. Second, it is conceptual in nature, with a theoretical basis that provides benefits for the solution of MDO problems. The framework provides a platform for solving the multidisciplinary analysis (MDA) problem and computing coupled derivatives automatically and in a general way, using any existing method or combination of methods. Finally, it is highly invasive, when used in a manner that takes advantage of these tools. The framework is more useful when components are smaller units of code rather than large, 'black-box' codes; however, to facilitate this, the framework allows components to be defined in a simple way and automates specification of dependencies between components.

A. Theory

The most basic elements in an MDO framework are *variables* and *components*. We define the full problem — that is, the set of all variables and components — as a *multidisciplinary system*. A variable is an object containing one or more units of data of some type. Variables are either independent, meaning set by the user, or dependent, meaning computed by a component. Here, we assume variables are either real or complex and not integer or binary, consistent with the earlier premise that all components are assumed to be differentiable.

A component often, but not always, represents a discipline. Its defining characteristic is that it is a unit of code that produces some output variables which are physically related to each other in some way. In terms of software, components are represented by a single class or function at the highest level, but conceptually, the boundaries between

components are somewhat arbitrary. For example, a finite element solver may consist of one component that computes displacements and another for the stresses, or they may be combined into a single component that computes stresses, assuming the displacements are not needed elsewhere. Mathematically, variables can be thought of as elements of vector spaces and components can be thought of as vector-valued functions.

CONVENTIONAL APPROACH: *Multidisciplinary analysis* refers to the process by which all dependent variables are computed, so that the state of the multidisciplinary system is known. The conventional thinking views evaluation of a system with multiple components as a sequential process, with each computing outputs that can be passed to other components as inputs. Execution of the components is performed in a particular order and computed data is immediately made available to other components in a manner analogous to the Gauss-Seidel method for solving linear systems.

NEW APPROACH: For reasons that will become clear later, the new approach removes the notion that components compute outputs which are then passed to other components as inputs, and execution of components is no longer necessarily sequential. Components have arguments similar to the traditional approach, but the meaning of their output is different. A component is once again a vector-valued function of variables, $\boldsymbol{v}$, but it is called *converged* when the output of the function is the zero vector. These output values are called constraints and denoted by $\boldsymbol{C}_i$ for the i^{th} component. Each component is a function of the same vector $\boldsymbol{v}$, and the size of $\boldsymbol{v}$ is equal to the sum of the sizes of the $\boldsymbol{C}_i$. A key conceptual difference is that variables are not owned by components that compute them and then pass them around; rather, it is more appropriate to think of all variables as belonging to the framework and are simply read by components but not actually controlled by any component. Here, multidisciplinary analysis is the process by which all constraints are simultaneously driven to zero, achieving a converged system.

MATHEMATICAL FORMULATION: It will become obvious that the new approach generalizes the conventional thinking and can handle any type of problem.

All variables in any multidisciplinary problem can be classified as one of three types:

1. Independent variables: $\boldsymbol{x}$

2. Explicitly defined variables: $\bar{\boldsymbol{y}}_i = \bar{\boldsymbol{Y}}_i(\boldsymbol{x}, \bar{\boldsymbol{y}}_{j\neq i}, \boldsymbol{y}_k)$
$$\implies \begin{pmatrix} \bar{\boldsymbol{y}}_1 \\ \vdots \\ \bar{\boldsymbol{y}}_{\bar{N}} \end{pmatrix} = \begin{pmatrix} \bar{\boldsymbol{Y}}_1(\boldsymbol{x}, \bar{\boldsymbol{y}}_2, \ldots, \bar{\boldsymbol{y}}_{\bar{N}}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) \\ \vdots \\ \bar{\boldsymbol{Y}}_{\bar{N}}(\boldsymbol{x}, \bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_{\bar{N}-1}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) \end{pmatrix}$$

3. Implicitly defined variables: $\boldsymbol{y}_i : \boldsymbol{R}_i(\boldsymbol{x}, \bar{\boldsymbol{y}}_j, \boldsymbol{y}_{k\neq i}, \boldsymbol{y}_i) = \boldsymbol{0}$
$$\implies \begin{pmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_N \end{pmatrix} : \begin{pmatrix} \boldsymbol{R}_1(\boldsymbol{x}, \bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_{\bar{N}}, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N, \boldsymbol{y}_1) \\ \vdots \\ \boldsymbol{R}_N(\boldsymbol{x}, \bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_{\bar{N}}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N-1}, \boldsymbol{y}_N) \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{pmatrix}$$

Above, each quantity is a vector, the bar indicates that the variable is explicitly defined, the subscript refers to the component index, and in the shorthand notation, a variable with a subscript in the argument of a function indicates that it is an argument for all values of the subscript. $\bar{N}$ and N are the total numbers of explicitly and implicitly defined variables, respectively.

Suppose we are given x^*, along with $\bar{\boldsymbol{y}}_i^*$ and $\boldsymbol{y}_i^*$ such that $\bar{\boldsymbol{y}}_i^* = \bar{\boldsymbol{Y}}_i(\boldsymbol{x}^*, \bar{\boldsymbol{y}}_{j\neq i}^*, \boldsymbol{y}_k^*)$ and $\boldsymbol{R}_k(\boldsymbol{x}^*, \bar{\boldsymbol{y}}_j^*, \boldsymbol{y}_{k\neq i}^*, \boldsymbol{y}_i^*) = \boldsymbol{0}$. Let us construct a vector of variables and a vector-valued function,

$$\boldsymbol{v} = \begin{pmatrix} \boldsymbol{x} \\ \bar{\boldsymbol{y}}_1 \\ \vdots \\ \bar{\boldsymbol{y}}_{\bar{N}} \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_N \end{pmatrix} \quad \text{and} \quad \boldsymbol{C}(\boldsymbol{v}) = \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^* \\ \bar{\boldsymbol{y}}_1 - \bar{\boldsymbol{Y}}_1(\boldsymbol{x}, \bar{\boldsymbol{y}}_2, \ldots, \bar{\boldsymbol{y}}_{\bar{N}}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) \\ \vdots \\ \bar{\boldsymbol{y}}_{\bar{N}} - \bar{\boldsymbol{Y}}_{\bar{N}}(\boldsymbol{x}, \bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_{\bar{N}-1}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) \\ -\boldsymbol{R}_1(\boldsymbol{x}, \bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_{\bar{N}}, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N, \boldsymbol{y}_1) \\ \vdots \\ -\boldsymbol{R}_N(\boldsymbol{x}, \bar{\boldsymbol{y}}_1, \ldots, \bar{\boldsymbol{y}}_{\bar{N}}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N-1}, \boldsymbol{y}_N) \end{pmatrix}, \tag{1}$$

where all quantities are vectors once again.

Based on these definitions, we have $\boldsymbol{C}(\boldsymbol{x}^*, \bar{\boldsymbol{y}}^*, \boldsymbol{y}^*) = 0$, and $\boldsymbol{C}(\boldsymbol{v}) = 0$ forms a nonlinear system of equations whose solution is the solution of the multidisciplinary system. Equation (1) represents the mathematical formulation of the general multidisciplinary analysis problem.

GENERAL EQUATION FOR COUPLED DERIVATIVES If we define $\boldsymbol{y}$ and $\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{y})$ by concatenating the appropriate sub-vectors in Eq. (1), we can write,

$$\boldsymbol{v} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \quad , \quad \boldsymbol{C}(\boldsymbol{v}) = \begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}^* \\ \boldsymbol{R}(\boldsymbol{x}, \boldsymbol{y}) \end{pmatrix}. \tag{2}$$

In this simplified form, the Jacobian of partial derivatives is

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{v}} = \begin{bmatrix} \boldsymbol{\mathcal{I}} & \boldsymbol{0} \\ \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{y}} \end{bmatrix}. \tag{3}$$

Since $det(\partial \boldsymbol{C}/\partial \boldsymbol{v}) = det(\boldsymbol{\mathcal{I}}) det(\partial \boldsymbol{R}/\partial \boldsymbol{y}) \neq 0$, $\partial \boldsymbol{C}/\partial \boldsymbol{v}$ is invertible. Given this fact, the inverse function theorem tells us that if $\boldsymbol{C}$ is smooth on an open set, there exists a locally-defined inverse function whose Jacobian is the inverse of $\partial \boldsymbol{C}/\partial \boldsymbol{v}$. That is, $[\partial \boldsymbol{C}/\partial \boldsymbol{v}]^{-1} = d\boldsymbol{v}/d\boldsymbol{c}$, and this allows us to compute total derivatives. We now have the result derived in [10]:

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{v}} \frac{d\boldsymbol{v}}{d\boldsymbol{c}} = \boldsymbol{\mathcal{I}} = \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{v}}^T \frac{d\boldsymbol{v}}{d\boldsymbol{c}}^T. \tag{4}$$

This equation contains as special cases the chain rule, algorithmic differentiation, the direct and adjoint methods, the coupled analytic methods, and the chain rule for coupled multidisciplinary systems, also known as coupled functional equations and GSE2 in the literature.

OUTCOME Based on the formulation presented in this section, performing multidisciplinary analysis and optimization essentially reduces to solving three types of problems — the coupled system (using Newton's method) and the forward and reverse modes for computing derivatives. Thus, we only need to concern ourselves with how to compute $\boldsymbol{C}$, $\partial \boldsymbol{C}/\partial \boldsymbol{v}$, and $[\partial \boldsymbol{C}/\partial \boldsymbol{v}]^T$, as the mathematics reduce to three linear systems:

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{v}} \Delta \boldsymbol{v} = -\boldsymbol{C} \tag{5}$$

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{v}} \frac{d\boldsymbol{v}}{d\boldsymbol{c}} = \boldsymbol{\mathcal{I}} \tag{6}$$

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{v}}^T \frac{d\boldsymbol{v}}{d\boldsymbol{c}}^T = \boldsymbol{\mathcal{I}}. \tag{7}$$

B. Solver

The types of problems we are interested in solving generally involve thousands or possibly millions of degrees of freedom. At this scale, nonlinear and linear systems must be represented and solved using very specific methods to keep memory and computing costs manageable. The framework must be general enough to be compatible with these methods, adding stringent requirements to its design.

In particular, the framework must be able to handle a great deal of heterogeneity among the components. In terms of the nonlinear problem, a component may compute variables which are nonlinearly related to each other and to variables from other components, but are sequential. An example is explicit time-marching. The fourth-order Runge Kutta scheme can be used to solve a nonlinear ODE efficiently because all state variables depend only on those of previous time instances, and formulating this problem as a nonlinear system and solving it using Newton's method significantly increases computation cost unnecessarily. When components compute variables that can be computed sequentially, the framework should be able to take advantage of this in an elegant way.

In terms of the linear system, some components may already have preconditioners or factorizations for the square Jacobian of its constraints with respect to its state variables. For instance, domain decomposition-based preconditioning is often used in computational fluid dynamics (CFD) to solve the linearized system for the fluid variables more efficiently. In computational structural mechanics (CSM), the stiffness matrix is often factorized and stored in a sparse format because of the high condition numbers that are frequently encountered and the fact that the system is often linear. A multidisciplinary system that combines CFD and CSM would be more efficient if it could use the existing preconditioner and factorization, but the framework must be designed to handle these elegantly. There are many different cases for the storage of the Jacobian as well, including dense, sparse, and matrix-free.

In addition, the nonlinear and linear systems can be solved at varying levels of decomposition. For both types of systems, solving one component at a time and repeating until the full system is converged corresponds to the block Jacobi or block Gauss-Seidel methods, depending on whether newly computed variables are updated immediately or at the end of each iteration. There can be arbitrary levels of nesting as a group of components that are solved together can have sub-groups that are solved together as well.

For multidisciplinary analysis, the solution that satisfies all of these requirements is a modified version of Newton's method. We add an additional block solve step at the start of every outer iteration and the linear system is solved using an iterative method with the matrix and preconditioner represented as linear operators in the general case. There are seven operations at the multidisciplinary system level, each of which calls its counterparts at the component level, and these are discussed below.

A. SOLVE: At the beginning of each iteration of Newton's method, each component's solve method is run. At the component level, the solve method attempts to achieve convergence of its own variables, with all variables from other components fixed at the most recent value. This method is optional and can be exact or simply provide an approximate solution. The solve method should typically be provided when the system is triangular as in the case of explicit time-marching, when efficient code for solving the single-discipline system already exists, or when the Jacobians are not available and nonlinear block Gauss-Seidel or block Jacobi are the only options.

B. EVALUATE: With the new set of values for the variables, each component's evaluate method is called to obtain the full $\boldsymbol{C}$ vector, to be used both as a convergence criterion and on the right hand side of the Newton system.

C. LINEARIZE: This is another optional method that is only implemented for certain types of components. If applicable, components assemble the Jacobian and form any full or incomplete factorizations they may use during the solution of the Newton system.

D. LINEAR SOLVE The final step in the outer (nonlinear) loop is the linear solve using a preconditioned iterative method. In this paper, the generalized minimual residual method (GMRES) is used for all problems as it has shown good performance. Right preconditioning is used for the matrix $\boldsymbol{J} = \partial \boldsymbol{C}/\partial \boldsymbol{v}$, with $\boldsymbol{M} \approx \boldsymbol{J}$ as the preconditioner. In practice, each component preconditions its own variables, so calling them in sequence or in parallel gives the effect of applying a global preconditioner that is block diagonal with the i^{th} block representing the i^{th} component's preconditioner.

For both the Jacobian $\boldsymbol{J}$ and the matrix $\boldsymbol{M}^{-1}$, the framework treats them as linear operators. Each component must have *applyJ* and *applyMinv* methods that accept the global vector representing the argument, apply the matrix-vector product for the elements in the global variable vector that are relevant, and return another global vector with the appropriate entries written. Through this approach, the framework treats dense, sparse, and matrix-free cases in a consistent manner and the component's linear operator method uses whichever matrix representation it has and simply returns the matrix-vector product regardless of how it is computed. The same approach is used to solve the forward and reverse mode derivative equations — the preconditioned linear system is solved using an iterative method. In the reverse mode, the matrix is transposed, so the relevant methods are *applyJT* and *applyMinvT*.

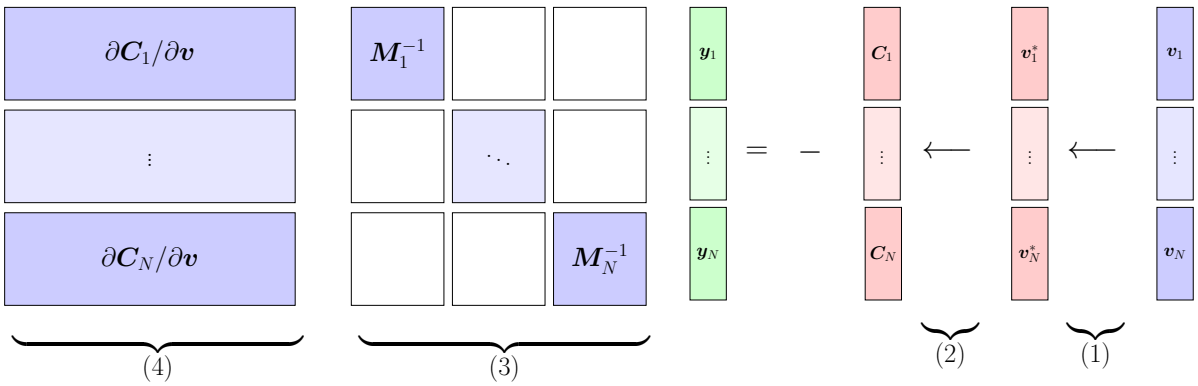
Table 1 shows how this formulation of the multidisciplinary analysis problem generalizes a wide variety of types of problems, and the proposed solver naturally becomes one of the existing methods depending on which methods are provided by the components. Figure 1 provides a visual illustration of the Newton and coupled analytic linear systems.

C. Software Design

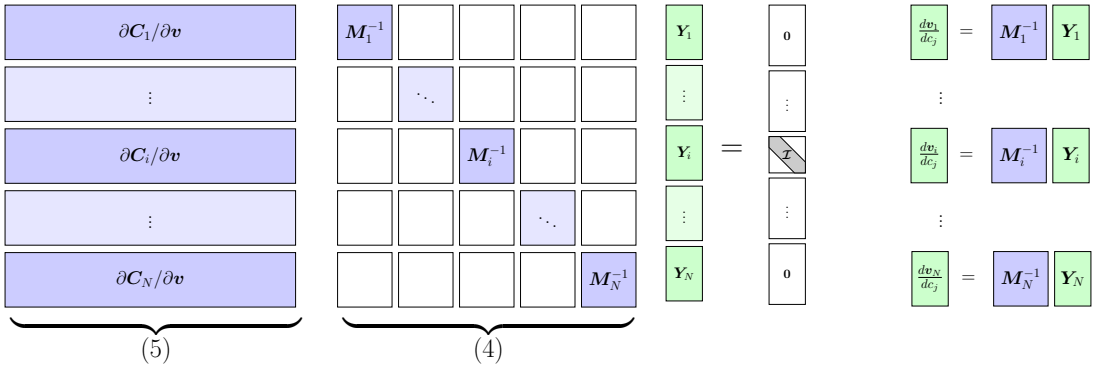
The framework has been implemented in Python using only the *NumPy* and *Scipy* libraries. Because of the conceptual simplicity of the framework, the entire implementation is contained within a single Python module that is on the order of only a few hundred lines of code.

In this implementation, there are three classes: the *Problem*, *Component*, and *Variable* classes. The *Problem* class represents the full MDO problem and contains all component instances. The *Component* class is a base class, so any component the user defines must inherit from it and it must contain instances of all variables that it owns. There are eight required methods for the component class — the seven discussed in the previous section and an initialize method:

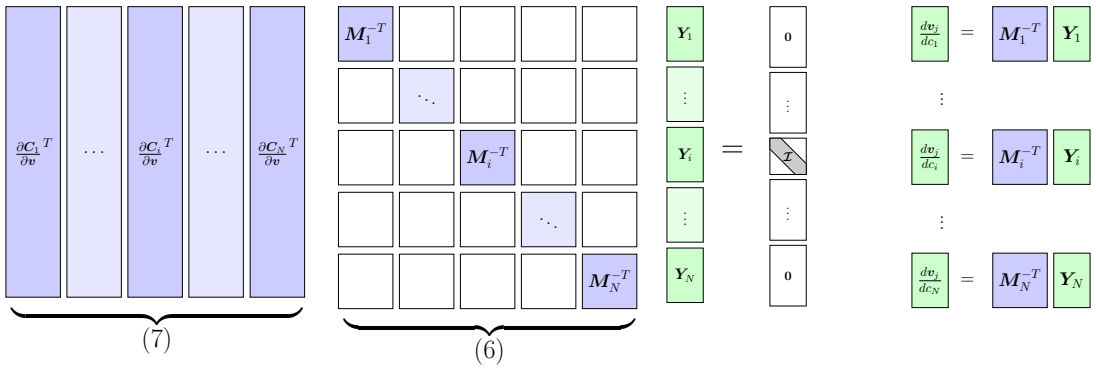
- *initialize*: define the variables associated with the component, along with their sizes and initial values
- *linearize*: optional, pre-compute the Jacobian



$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{v}} \Delta \boldsymbol{v} = -\boldsymbol{C}$$



$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{v}} \frac{d\boldsymbol{v}}{d\boldsymbol{c}} = \boldsymbol{\mathcal{I}}$$



$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{v}}^T \frac{d\boldsymbol{v}}{d\boldsymbol{c}}^T = \boldsymbol{\mathcal{I}}$$

Figure 1: Visual description of the nonlinear solver and coupled analytic systems in the forward and reverse modes. Subscripts indicate the component to which the vector $\boldsymbol{C}$ or $\boldsymbol{v}$ belongs. There are seven operations: (1) block Gauss-Seidel or block Jacobi solves within each discipline, (2) evaluation of all residuals, (3) linearization of each component, (4) preconditioner for the Jacobian, (5) the Jacobian of constraints with respect to variables, (6) the preconditioner for the transposed Jacobian, and (7) the transposed Jacobian.

Method	*Jacobian* (4)	*Preconditioner* (3)	*solve* (1)
Nonlin. block Gauss-Seidel	Identity	Identity	Exact solve
Nonlin. block Jacobi	Identity	Identity	Exact, parallel
Linear block Jacobi	Exact	Exact	No action
Coupled Newton's method	Exact	Approximate	No action
Component	*Jacobian* (4)	*Preconditioner* (3)	*Local solve* (1)
Triangular	Exact	Back subst.	Exact solve
Preconditioned	Exact	Preconditioner	No action
Factorized	Exact	Exact inverse	No action
Jacobian-free	Directional derivative		No action

Table 1: Listing of a number of types of problems and solution methods which are captured by the proposed solver. By implementing the appropriate type of solve or linear operator (e.g. no action, identity operator, etc.), the solver effectively becomes one of the listed methods. The numbers refer to Fig. 1.

- *solve*: optional, local solve for the 'owned' entires of $\boldsymbol{v}$
- *execute*: compute 'owned' entries of $\boldsymbol{C}$
- *applyJ*: optional, Jacobian-vector product
- *applyJT*: optional, Jacobian transpose-vector product
- *applyMinv*: optional, preconditioner
- *applyMinvT*: optional, transposed preconditioner

One of the implementation details that merits discussion is the manner in which data is read and written. Since a single linear solver is applied to the monolithic problem, the full set of variables must be agglomerated into a single global vector. The same is the case for the constraints and the vector of unknowns for the Newton and coupled analytic linear systems.

A feature that significantly improves ease of use is the ability to access variables by their names and in their proper shapes (arrays, if applicable, as opposed to flattened vectors). This is done without the memory and computation cost associated with keeping an additional local copy of the variable in the proper array shape by using *numpy.ndarray.view*. Instances of this class can access entries of an array while giving the appearance of having a different array shape, and this is done without copying the data in memory.

This provides a nice solution from a usability standpoint for codes that are run in serial or parallel with shared memory; however, it fails when data is distributed across multiple processors. The only solution in that case is to keep the variables stored locally, and abstract the global vector of variables. The global vector would not be stored explicitly; rather, it would have a method for computing dot products with other vectors. This is an avenue for future work as it would be necessary to implement an iterative solver that works with linear systems with neither the matrix nor the vector stored. This would be possible with GMRES, for instance, since only matrix-vector and vector-vector products are required, for building the Krylov subspace and for orthonormalizing the basis, respectively.

III. Modeling Tools

This section will describe a number of tools and techniques to aid in the solution of the CADRE multidisciplinary analysis and optimizations problems. These include software architecture, computation of derivatives, time-marching, a constraint aggregation function, and multidimensional interpolation.

A. Software Architecture

All of the codes used in modeling CADRE follow the structure detailed in the previous section. Each discipline is implemented using between one and ten components in the framework, and there is one additional component which contains the parameters — these represent the independent variables that can be chosen as design variables for an optimization problem.

Nearly every component is implemented in Fortran, but wrapped using Python. Since each one must be a Python class that inherits from the base *Component* class, the required Python-level methods such as *evaluate* and *linearize* call Fortran subroutines to carry out the computationally-intensive calculations as quickly as possible. Each discipline essentially has a Fortran library of subroutines that it can call from, and certain subroutines can be shared across disciplines — e.g. application of a rotation matrix to a vector. The Python-Fortran combination has proved to be highly effective; we can take advantage of both the efficiency of Fortran and the object-oriented, plotting, and scripting features of Python. This has enabled the development of a framework that can solve a problem with millions of degrees of freedom in seconds while maintaining simplicity and ease of use.

B. Computation of Derivatives

Though coupled derivatives are computed at the framework level, the Jacobian block $\partial \boldsymbol{C}_i / \partial \boldsymbol{v}$ is computed by the i^{th} component. Essentially, each component must be able to compute the derivatives of any variables it computes with respect to any variables on which it depends.

There are a number of options for computing derivatives. The simplest option is finite-differencing, either column-by-column or as a directional derivative. However, the accuracy of finite differences is limited because at small step sizes, subtractive cancellation error dominates, while at larger step sizes, the error term in the first order Taylor expansion becomes significant. The complex-step method avoids this issue since there is no subtraction operation in the formula; however, the cost of differentiation is proportional to the number of variables, which for this problem is at least in the tens of thousands. Algorithmic differentiation is another option with error theoretically on the order of machine precision, but the cost is proportional to either the number of outputs or number of inputs, which are both large numbers in this problem.

For these reasons, each component has been hand differentiated. This approach yields numerically exact derivatives and can be more efficient than any of the other methods. Furthermore, many of the Jacobians are large in size (tens of thousands by tens of thousands), but sparse. These sparse matrices are assembled directly in Fortran very efficiently.

C. Explicit Time-Marching

In total, there are five ODEs in time that must be integrated multiple times during the multidisciplinary analysis. These are in the orbit dynamics, attitude dynamics, battery, temperature, and communication disciplines. The most efficient method is to use explicit time-marching to solve these ODEs, and a high-order multi-stage method such as the fourth order Runge Kutta scheme (RK4) yields sufficient accuracy.

To reduce time and complexity, RK4 is implemented in a general way and both the Fortran subroutines and Python wrappers are reused in each instance. The time-dependent ODE for a state variable vector $\vec{y}$ is

$$\dot{\vec{y}} = \vec{f}(t, \vec{x}, \vec{y}), \tag{8}$$

where the time derivative is in general a nonlinear function of time, external variables $\vec{x}$, and the state variable itself.

The RK4 scheme is given below, written as a constraint so that it fits into the MDO framework present earlier.

$$\begin{aligned} R_{i+1,k}(\vec{y}) &= y_{i+1,k} - y_{i,k} - \frac{\Delta t}{6}(a_k + 2b_k + 2c_k + d_k) \\ a_k &= f_k(t_i, \vec{x}_i, \vec{y}_i) \\ b_k &= f_k(t_i + \tfrac{1}{2}\Delta t, \vec{x}_i, \vec{y}_i + \tfrac{1}{2}\Delta t\ \vec{a}) \\ c_k &= f_k(t_i + \tfrac{1}{2}\Delta t, \vec{x}_i, \vec{y}_i + \tfrac{1}{2}\Delta t\ \vec{b}) \\ d_k &= f_k(t_i + \Delta t, \vec{x}_i, \vec{y}_i + \Delta t\ \vec{c}) \end{aligned} \tag{9}$$

We need to be able to compute derivatives of $\vec{R}$ with respect to both the state variables $\vec{y}$ and the external variables $\vec{x}$ at each instance of time. The equations below are derived independent of the ODE, so they can be implemented in a general way and reused for each component. Given the ODE-specific derivatives $\frac{\partial f_k}{\partial y_{i,j}}$ and $\frac{\partial f_k}{\partial x_{i,j}}$, the following equations can be used to compute the full Jacobian for the component:

$$\frac{\partial R_{i+1,k}}{\partial y_{i,j}} = \frac{\partial y_{i+1,k}}{\partial y_{i,j}} - \delta_{k,j} - \frac{\Delta t}{6}\left(\frac{\partial a_k}{\partial y_{i,j}} + 2\frac{\partial b_k}{\partial y_{i,j}} + 2\frac{\partial c_k}{\partial y_{i,j}} + \frac{\partial d_k}{\partial y_{i,j}}\right) \tag{10}$$

$$\frac{\partial R_{i+1,k}}{\partial x_{i,j}} = \frac{\partial y_{i+1,k}}{\partial x_{i,j}} - \frac{\Delta t}{6}\left(\frac{\partial a_k}{\partial x_{i,j}} + 2\frac{\partial b_k}{\partial x_{i,j}} + 2\frac{\partial c_k}{\partial x_{i,j}} + \frac{\partial d_k}{\partial x_{i,j}}\right) \tag{11}$$

$$\frac{\partial a_k}{\partial y_{i,j}} = \left.\frac{\partial f_k}{\partial y_{i,j}}\right|_{a_k} \quad (12) \qquad \frac{\partial a_k}{\partial x_{i,j}} = \left.\frac{\partial f_k}{\partial x_{i,j}}\right|_{a_k} \quad (16)$$

$$\frac{\partial b_k}{\partial y_{i,j}} = \left.\frac{\partial f_k}{\partial y_{i,j}}\right|_{b_k} + \frac{\Delta t}{2}\left.\frac{\partial f_k}{\partial y_{i,l}}\right|_{b_k}\frac{\partial a_l}{\partial y_{i,j}} \quad (13) \qquad \frac{\partial b_k}{\partial x_{i,j}} = \left.\frac{\partial f_k}{\partial x_{i,j}}\right|_{b_k} + \frac{\Delta t}{2}\left.\frac{\partial f_k}{\partial y_{i,l}}\right|_{b_k}\frac{\partial a_l}{\partial x_{i,j}} \quad (17)$$

$$\frac{\partial c_k}{\partial y_{i,j}} = \left.\frac{\partial f_k}{\partial y_{i,j}}\right|_{c_k} + \frac{\Delta t}{2}\left.\frac{\partial f_k}{\partial y_{i,l}}\right|_{c_k}\frac{\partial c_l}{\partial y_{i,j}} \quad (14) \qquad \frac{\partial c_k}{\partial x_{i,j}} = \left.\frac{\partial f_k}{\partial x_{i,j}}\right|_{c_k} + \frac{\Delta t}{2}\left.\frac{\partial f_k}{\partial y_{i,l}}\right|_{c_k}\frac{\partial c_l}{\partial x_{i,j}} \quad (18)$$

$$\frac{\partial d_k}{\partial y_{i,j}} = \left.\frac{\partial f_k}{\partial y_{i,j}}\right|_{d_k} + \Delta t\left.\frac{\partial f_k}{\partial y_{i,l}}\right|_{d_k}\frac{\partial d_l}{\partial y_{i,j}} \quad (15) \qquad \frac{\partial d_k}{\partial x_{i,j}} = \left.\frac{\partial f_k}{\partial x_{i,j}}\right|_{d_k} + \Delta t\left.\frac{\partial f_k}{\partial y_{i,l}}\right|_{d_k}\frac{\partial d_l}{\partial x_{i,j}}. \quad (19)$$

D. Kreisselmeier-Steinhauser Function

The coupled derivative equations are very efficient because they give either the full vector of derivatives with respect to a single variable (forward mode) or the full gradient of a variable (reverse mode) using a single linear solve. Since we plan to perform optimization with a large number of design variables, the reverse mode must be used, but it requires a linear solve for each constraint in the optimization problem. However, the battery discipline requires four inequality constraints at each time instance — maximum charge rate, maximum discharge rate, minimum state of charge, maximum state of charge — which results in tens of thousands of constraints.

The solution is to use the Kreisselmeier-Steinhauser (KS) function to aggregate constraints over all the time instances into a single criterion. The KS function is given by

$$KS(x) = f_{i,max}(x) + \frac{1}{\rho}\ln\sum_i e^{\rho(f_i(x)-f_{i,max}(x))}, \quad (20)$$

where f_i is the i^{th} function is the vector of functions we wish to aggregate, i, max is the index of the function with the largest value at x, and ρ is a parameter that is problem-dependent. In the limit as ρ approaches infinity, the KS function approaches the maximum function as $e^{\rho\cdot 0}$ dominates in the sum, and $KS(x)$ approaches $f_{i,max}(x)$. For finite ρ, the KS function is a smooth function that is dominated by the f_i with the largest values; thus, as an inequality constraint, the KS function encourages the optimizer to resolve the largest infeasibilities first and eventually choose a point at which the KS function itself satisfies the inequality constraint.

E. Multi-dimensional Interpolation

Multi-dimensional interpolation is useful when there is only a table of data available for a calculation, when there are non-differentiable points that must be smoothed, or when a particular model is too complicated or expensive to implement. Typically, there is a structured array of data where the rank of the array is the number of variables upon which the quantity of interest depends.

For this purpose, an n-dimensional tensor-product B-spline interpolant has been implemented. B-splines are polynomial splines of minimal support that are flexible in terms of the number of control points and order. Tensor products of B-splines in multiple parametric directions form multi-dimensional interpolants.

To illustrate, a single dimensional B-spline is of the form,

$$\vec{Q} = B_j(u)\vec{C}_j \quad (21)$$

$$Q_{i,k} = B_{i,j}C_{j,k}, \quad (22)$$

where $\vec{Q} = Q_k$ is the quantity of interest and $\vec{C}_j = C_{j,k}$ are the control points, and the second equation shows the discretized form involving matrix multiplication. It is implied that the basis functions have been evaluated at the parametric coordinates u_i. The reader is encouraged to refer to [11] for more information on B-splines.

IV. Disciplines

This section presents all of the disciplines for which models of CADRE have been implemented. The disciplines include orbit dynamics, exposed area, attitude dynamics and control, Sun position, thermal, electrical, battery, and communication.

For vectors, the nomenclature used in this section and the communication section is as follows. Upper-case subscripts distinguish frames of reference, with B, R, E, and I representing body-fixed frame, rolled body-fixed frame

(explained later), Earth-fixed frame, and Earth-centered inertial frame, respectively. Lower-case subscripts represent the origins of frames — b, e, g, and s denote body (satellite), Earth, ground station, and Sun, respectively. For instance, $\hat{r}_{b/e}$ signifies the vector pointing from the earth's origin to the satellite's origin. The axes of the body-fixed frame are denoted $\hat{i}_B$, $\hat{j}_B$, and $\hat{k}_B$. Orientation matrices are represented with the letter O, so for instance, $O_{B/I}$ represents the rotation from the Earth-centered inertial frame to the body-fixed frame.

A. Orbit Dynamics

The orbit dynamics discipline computes the earth-to-body position vector in the Earth-centered inertial (ECI) frame. In the orbit equation, there are terms that represent the fact that the Earth is not a perfectly spherical and homogeneous mass. These are captured in the J_2, J_3, and J_4 coefficients in the following equation:

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r} - \frac{3\mu J_2 R_e^2}{2r^5}\left[\left(1-\frac{5r_z^2}{r^2}\right)\vec{r} + 2r_z\hat{z}\right] \tag{23}$$

$$- \frac{5\mu J_3 R_e^3}{2r^7}\left[\left(3r_z - \frac{7r_z^3}{r^2}\right)\vec{r} + \left(3r_z - \frac{3r^2}{5r_z}\right)r_z\hat{z}\right] \tag{24}$$

$$+ \frac{15\mu J_4 R_e^4}{8r^7}\left[\left(1-\frac{14r_z^2}{r^2}+\frac{21r_z^4}{r^4}\right)\vec{r} + \left(4-\frac{28r_z^2}{3r^2}\right)r_z\hat{z}\right]. \tag{25}$$

The J_2, J_3, and J_4 terms must be considered because their effect is to rotate the orbit plane on the scale of months. If they were to be ignored, a fin angle that may maximize power generation initially may no longer be optimal after the orbit plane has changed. This also makes the CADRE design problem a multi-scale problem in time because much of the system's behavior occurs on the scale of minutes and hours, since the period of the satellite's orbit is roughly 90 minutes. On the other hand, the slow rotation of the orbit plane affects power generation and communication as well, since the satellite's trajectory affects how much data can be transmitted, as it passes over ground stations.

B. Exposed Area

The CADRE CubeSat has 12 solar panels and 7 cells per panel. Four of the panels cover the sides of the body of the satellite, while the remaining eight panel cover the front and back sides of the four fins, as shown in Fig. 2. The four fins are set at the same angle relative to the body, but this angle can be changed with negligible effects on disciplines that are not modeled in this problem, so it is a natural choice as a design variable.

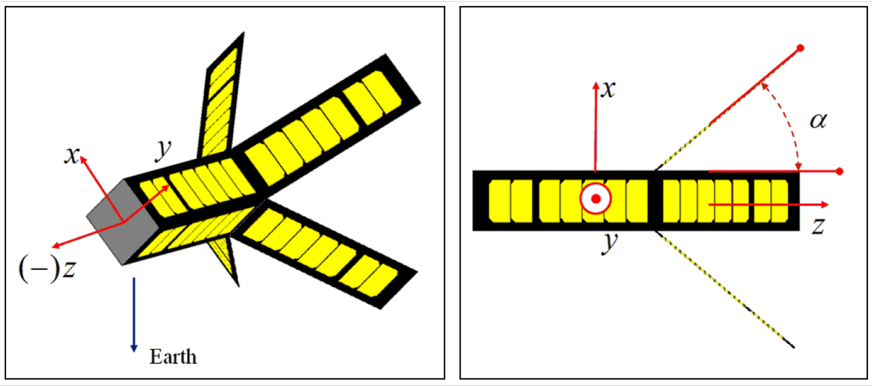
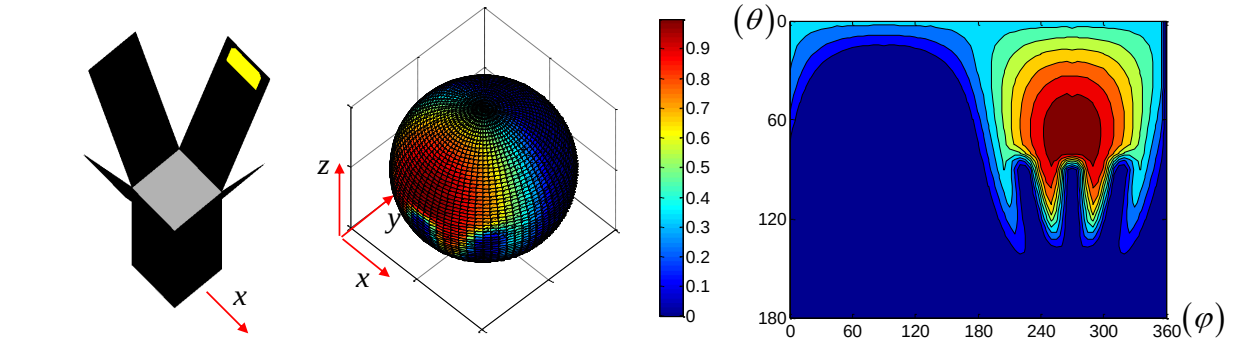


Figure 2: Rendering of the CADRE CubeSat showing the solar panels, cells, and fin angle.

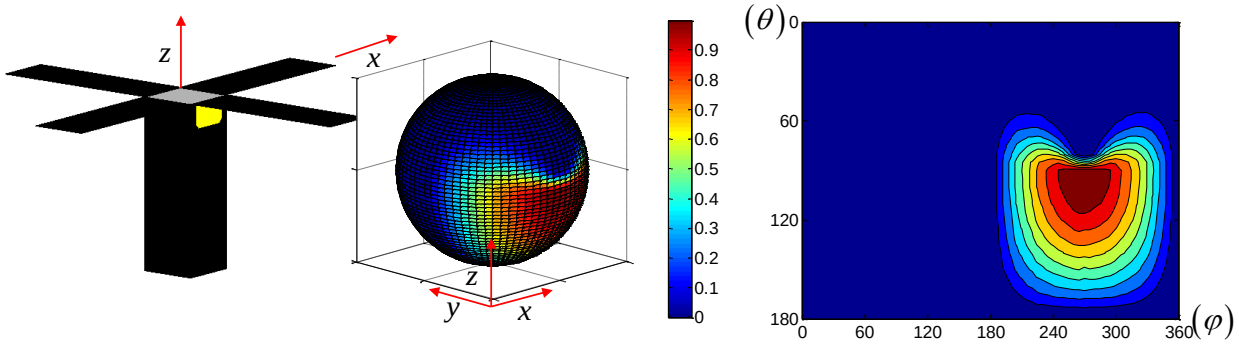
The exposed area discipline models the amount of each solar cell's area that is exposed to the Sun, projected onto the plane normal to the Sun's incidence. The 84 exposed areas depend on the fin angle, as well as the azimuth and elevation angles of the Sun in the body-fixed frame. These are computed using an OpenGL model of the geometry, with the fins discretized into small rectangles.

Since this model is both discontinuous and difficult to incorporate into the framework, a table of data is generated and the B-spline multi-dimensional interpolant is used as an approximation for the exposed areas in terms of the three parameters. This also has the effect of smoothing the areas as the scale at which the areas 'jump', which is related to the area of a pixel in the OpenGL model, is much lower than the distance between control points in the B-spline

interpolant. The interpolated function does not have sufficient precision to be affected by the discontinuous jumps, but it does have the degrees of freedom to follow the general trends. The scale of the discontinuities is evident from Fig. 3.



(a) Fin angle : 20°



(b) Fin angle : 90°

Figure 3: Representative plots of normalized exposed area as functions of relative Sun position.

C. Attitude Dynamics and Control

Based on the requirements of CADRE's science mission, it must always have a forward facing orientation, though the roll angle, γ, can change. Thus, the appproach used to model the attitude dynamics of the satellite is to compute the rotation matrix that yields this orientation, the angular velocity vector, and then the torque from the actuator to match the atttitude profile. Since we are modeling all time instances simultaneously, this approach essentially determines the control inputs using optimization instead of a controller.

The scientific instruments are mounted on the face in the $-\hat{k}_B$ direction in CADRE's body frame. Thus, we require that

$$\hat{k}_B = -\hat{v}_{b/e}. \tag{26}$$

If we choose to define the $\hat{j}_B$ direction to be parallel to the Earth-to-body vector, it follows that

$$\hat{j}_B = \hat{r}_{b/e} \quad \text{and} \quad \hat{i}_B = \hat{j}_B \times \hat{k}_B. \tag{27}$$

Instead of a direct transformation from the inertial frame to the body-fixed frame, there are two rotations with an intermediate frame that we will refer to here as the rolled body-fixed frame, denoted by R. This intermediate frame is the one that results from rotating the inertial frame to ensure that the $-\hat{k}_B$ direction is parallel to CADRE's velocity vector, and a second rotation implements the roll design variable, γ. The rotation matrices for the inertial to rolled frame and rolled to body-fixed frame are

$$O_{R/I} = \begin{bmatrix} \hat{i}_B^T \\ \hat{j}_B^T \\ \hat{k}_B^T \end{bmatrix} \quad \text{and} \quad O_{B/R} = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{28}$$

Using these matrices, we can find $O_{B/I}$, which yields angular velocity through the equation,

$$\begin{bmatrix} 0 & \omega_{B,3} & -\omega_{B,2} \\ -\omega_{B,3} & 0 & \omega_{B,1} \\ \omega_{B,2} & -\omega_{B,1} & 0 \end{bmatrix} = \dot{O}_{B/I} \cdot O_{B/I}^T, \tag{29}$$

where $\dot{O}_{B/I}$ is computed using the central difference formula.

By differentiating the expression for total angular momentum, we can get Euler's equations and an expression for the external torque in terms of the reaction wheel's angular velocity:

$$\vec{L} = J_B \cdot \vec{\omega}_B + J_{RW} \cdot \vec{\omega}_{RW} \tag{30}$$

$$\implies \dot{\vec{L}} = J_B \cdot \dot{\vec{\omega}}_B + \vec{\omega}_B \times (J_B \cdot \vec{\omega}_B) + J_{RW} \cdot \dot{\vec{\omega}}_{RW} + \vec{\omega}_B \times (J_{RW} \cdot \vec{\omega}_{RW}) = 0. \tag{31}$$

With the satellite's angular velocity, ω_B, known for each time instance, we can once again use the central difference formula to compute $\dot{\omega}_B$. Using these two quantities, we compute the required torque in the reaction wheel, integrate the attitude dynamics equation for the reaction wheel, then obtain the motor torque as follows:

$$\vec{\tau}_{RW} = J_B \cdot \dot{\vec{\omega}}_B + \vec{\omega}_B \times (J_B \cdot \vec{\omega}_B) \tag{32}$$

$$\dot{\vec{\omega}}_{RW} = -J_{RW}^{-1} \cdot [\vec{\tau}_{RW} + \vec{\omega}_B \times (J_{RW} \cdot \vec{\omega}_{RW})] \tag{33}$$

$$\vec{\tau}_m = -\vec{\tau}_{RW} - \vec{\omega}_B \times (J_{RW} \cdot \vec{\omega}_{RW}). \tag{34}$$

The current draw of the reaction wheel can then be computed with

$$I_{RW} = I_1 \omega^2 \frac{1}{\rho} \ln\left(e^{\rho\tau_m} + e^{-\rho\tau_m}\right) + I_2 \approx I_1 \omega^2 |\tau_m| + I_2, \tag{35}$$

where the absolute value function is approximated using the KS function to remove the non-smoothness at the origin. This simple formula captures the fact that more current is required to achieve a torque when the reaction wheel already has a large angular velocity, and the coefficients are determined from manufacturer's data.

D. Sun Position

The sun's position in the Earth-centered inertial (ECI) frame is computed based on the launch data and time from launch in Julian days. Using the inertial to body rotation matrix, the sun's position in the body frame is computed, which is then converted into spherical coordinates, azimuth and elevation, shown in Fig. 4. The equations for this conversion to spherical coordinates are

$$\begin{Bmatrix} r_x = r\sin\phi\cos\theta \\ r_y = r\sin\phi\sin\theta \\ r_z = r\cos\phi \end{Bmatrix} \implies \begin{Bmatrix} \theta = \arctan\left(\frac{r_y}{r_x}\right) \in [0, 2\pi) \\ \phi = \arccos\left(\frac{r_z}{r}\right) \end{Bmatrix}. \tag{36}$$

It is interesting to note that differentiating the expressions for θ and ϕ yields derivatives which are undefined at certain points. However, some algebraic manipulation after implicit differentiation yields

$$\begin{bmatrix} \frac{\partial\theta}{\partial r_x} & \frac{\partial\theta}{\partial r_y} & \frac{\partial\theta}{\partial r_z} \\ \frac{\partial\phi}{\partial r_x} & \frac{\partial\phi}{\partial r_y} & \frac{\partial\phi}{\partial r_z} \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -\csc\phi\sin\theta & \csc\phi\cos\theta & 0 \\ \cos\phi\cos\theta & \cos\phi\sin\theta & -\sin\phi \end{bmatrix}, \tag{37}$$

which is defined everywhere. The value r will never be zero since it represents the distance between the satellite and the center of the earth.

Another part of the sun position discipline is a component that checks if the satellite is in the earth's shadow [12]. To do this, the dot product, d, is computed between the position vectors of the satellite and sun in the ECI frame. If this value is positive, the satellite is closer to the sun than the earth, so the sun is definitely blocked from view. The norm of the cross product, c, between the same two vectors is also computed. If $d < 0$, we must also check that c, which is the normal distance between the satellite and the Sun-Earth position vector, is greater than the earth's radius, R_e, to ensure the satellite is receiving sunlight.

The line-of-sight variable, LOS, is essentially a multiplier for the exposed areas; it is 0 if the satellite is behind the earth and 1 otherwise. To smooth this discontinuous jump, we make the assumption that the sunlight does not

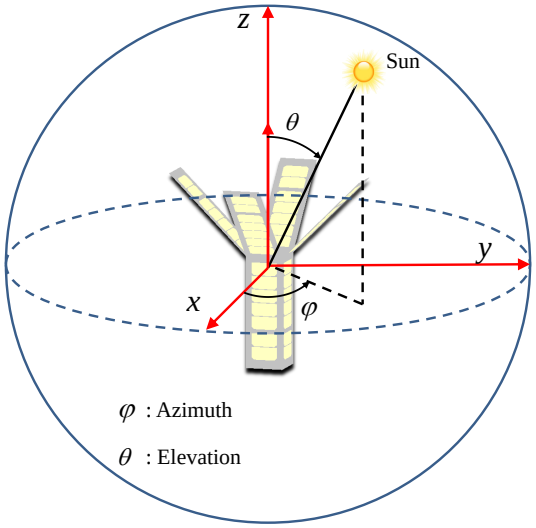


Figure 4: The description of the sun's position in the body-fixed frame in spherical coordinates.

decrease abruptly as the satellite moves into the earth's shadow, but smoothly transitions to zero. This is true to an extent because of the umbra and penumbra effect, but it is greatly exaggerated to avoid numerical difficulties in solving the optimization problem. Mathematically, a simple cubic function is constructed between $c = \alpha R_e$ and $c = R_e$, satisfying the C^0 and C^1 conditions at each end point. The value of α represents how far this smoothing effect extends into the earth's shadow and a typical value is $\alpha = 0.9$. The entire procedure used to compute LOS is illustrated in Fig. 5 and the equations are

$$d = \vec{r}_{b/e} \cdot \vec{r}_{s/e} \quad , \quad c = ||\vec{r}_{b/e} \times \vec{r}_{s/e}||_2 \quad , \quad \eta = \frac{c - \alpha R_e}{R_e - \alpha R_e} \tag{38}$$

$$LOS_s = \left\{ \begin{array}{ccc} 1 & , & d > 0 \\ 1 & , & c > R_e \\ 3\eta^2 - 2\eta^3 & , & \alpha R_e < c < R_e \\ 0 & , & c < \alpha R_e \end{array} \right\}. \tag{39}$$

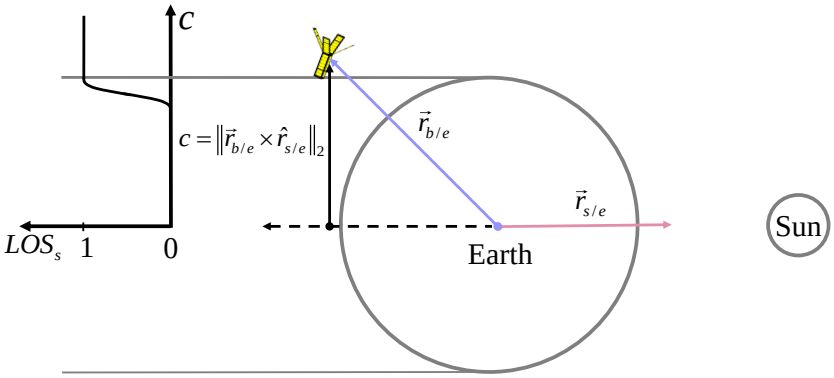


Figure 5: Illustration of the Sun line-of-sight variable.

E. Thermal

Temperature for the solar panels is an important consideration because receiving too much sunlight heats up the panels, reducing power generation and battery performance. A well designed and operated satellite can take advantage of cooling when it is behind the Earth, and generate as much power as possible without excessive heating.

Temperature is assumed to be uniform within each of the fins and the body. Thus, five values describe the temperature of the satellite at any given time, and both heating and radiation are considered from each of the 84 cells. The Stefan-Boltzmann law is used to model the rate of heat radiation, and the solar constant and exposed area are used to estimate the rate at which the cell is heated. The RK4 solver is used to integrate the nonlinear ODE represented by

$$\dot{T_k} = \sum_{p=1}^{12}\sum_{c=1}^{7} \delta(k,p)\frac{Q_{in,c,p} - Q_{out,c,p}}{m_k C_k} \tag{40}$$

$$Q_{in,c,p} = \alpha_{c,p} q A_{exp,c,p} LOS_s \tag{41}$$

$$Q_{out,c,p} = \epsilon_{c,p}\left(\frac{2\pi^5 k^4}{15c^2h^3}\right) T_k^4 A_T, \tag{42}$$

where m is the mass of the cell, C is the specific heat capacity, α is the absorptivity of the cell, ϵ is the emissivity of the cell, k is the Boltzmann constant, h is Planck's constant, c is the speed of light, A_T is the total area of the cell, and A_{exp} is the exposed area for the cell.

In some cases, it may be more productive not to install one or more of the cells and install a radiator using the same space instead. To add this to the optimization problem without adding a discrete variable, we linearly interpolate the cell and radiator properties and allow the optimizer the choose any $\xi \in [0,1]$ where $\xi = 0$ is a radiator, $\xi = 1$ is a cell, and intermediate values represent some weighted average of the two.

F. Electrical

The cells in a solar panel are connected in series, so their output voltages are added to compute the total voltage for the panel. The voltage is set, so as to maximize the power output, but the optimal voltage, and thus the optimal current, changes depending on the exposed area and temperature of the cells.

Each cell has a unique I-V curve which depends on its exposed area and temperature. The model [13] is a nonlinear implicit equation in I given by

$$I = I_{sc} - I_{sat}\exp\left\{\frac{V + R_s I}{V_T}\right\} - \frac{V + R_s I}{R_{sh}} \approx I_{sc} - I_{sat}\exp\left\{\frac{V}{V_T}\right\} \tag{43}$$

$$I_{sc} = LOS_s \frac{A_{exp}}{A_T} I_{sc0} \tag{44}$$

$$V_T = \frac{nkT}{q}, \tag{45}$$

where I_{sc0} is the short-circuit voltage assuming full exposure of the cell, $n = 1.35$, k is the Boltzmann constant, q is the charge of an electron, I_{sat} is the saturation current, R_s is the series resistance, and R_{sh} is the shunt resistance. By assuming R_s is small and neglecting the linear term, we obtain an approximation with which we can solve for V.

When the exposed area is low, the current may be greater than I_{sc}, which is the current at which $V = 0$, As current increases, voltage becomes negative and is modeled to asymptote to V_0, which is essentially a minimum voltage. By imposing C^0 and C^1 constraints, we can derive an expression for voltage in this range of currents, yielding the piecewise function,

$$V(I) = \begin{Bmatrix} V_T \ln\frac{I_{sc}+I_{sat}-I}{I_{sat}} & , & I <= I_{sc} \\ \frac{V_0^2 I_{sat}}{V_T(I - I_{sc} - \frac{V_0}{V_T} I_{sat}} & , & I > I_{sc} \end{Bmatrix}. \tag{46}$$

G. Battery

The battery's state of charge (SOC) is modeled using the ODE given by

$$\dot{SOC} = -\frac{\sigma}{24} SOC + \frac{P\eta}{VC'} \tag{47}$$

$$V = \left[a_3(SOC)^3 + a_2(SOC)^2 + a_1(SOC) + a_0\right]\left(2 - e^{\alpha\frac{T-T_0}{T_0}}\right), \tag{48}$$

where P is the power load on the battery, V is the voltage, η is the battery efficiency, C' is the nominal capacity, and σ represents the idle discharge rate. The battery resistance is assumed to be small, so V is simply the open-circuit voltage multiplied by an exponential term modeling the decreasing in voltage with increasing temperature. The previously mentioned Kreisselmeier-Steinhauser functions are used to aggregate charge, discharge, minimum SOC, and maximum SOC constraints over all time instances into one constraint for each.

H. Communication

The communication discipline models data transfer bit rate in terms of several variables. The approach used is to fix signal-to-noise ratio (SNR) to a minimum acceptable value to maintain a secure connection, and a line-of-sight variable, similar to that computed in the Sun position discipline is used to account for when the Earth is in the way. The equation [14] for bit rate is

$$B_r = \frac{c^2 G_r L_l}{16\pi^2 f^2 k T_s (SNR)} \frac{P_{comm} G_t}{S^2} LOS_c \tag{49}$$

$$\dot{D}_d = B_r, \tag{50}$$

where c is the speed of light, G_r is a constant representing receiver gain, L_l is a line loss factor, f is the frequency, k is the Boltzmann constant, T_s is the system noise temperature, P_{comm} is the power used to amplify the signal, G_t is the transmitter gain, and S is the distance between the ground station and the satellite. The total data downloaded, D_d, is computed by integrating the bit rate over time. The RK4 solver applied to this equation is equivalent to integration using Simpson's rule.

The LOS_c variable is computed based on the dot product between the normalized Earth-to-ground station vector and the Earth-to-body vector in the inertial frame. Once again, the discontinuous function is smoothed, in this case by assuming that the line-of-sight variable gradually increases as the satellite comes over the horizon. This is illustrated in Fig. 6.

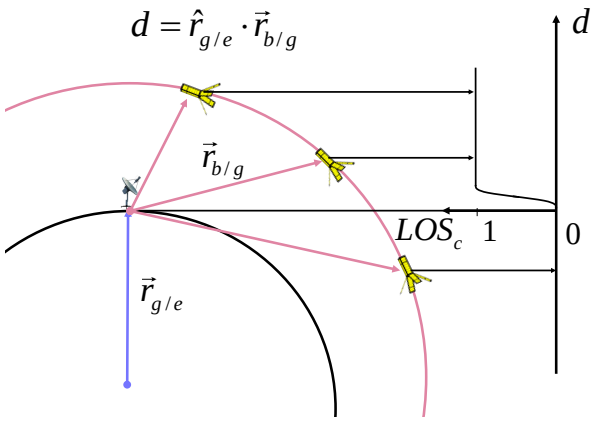


Figure 6: Illustration of the communication line-of-sight variable.

V. Optimization

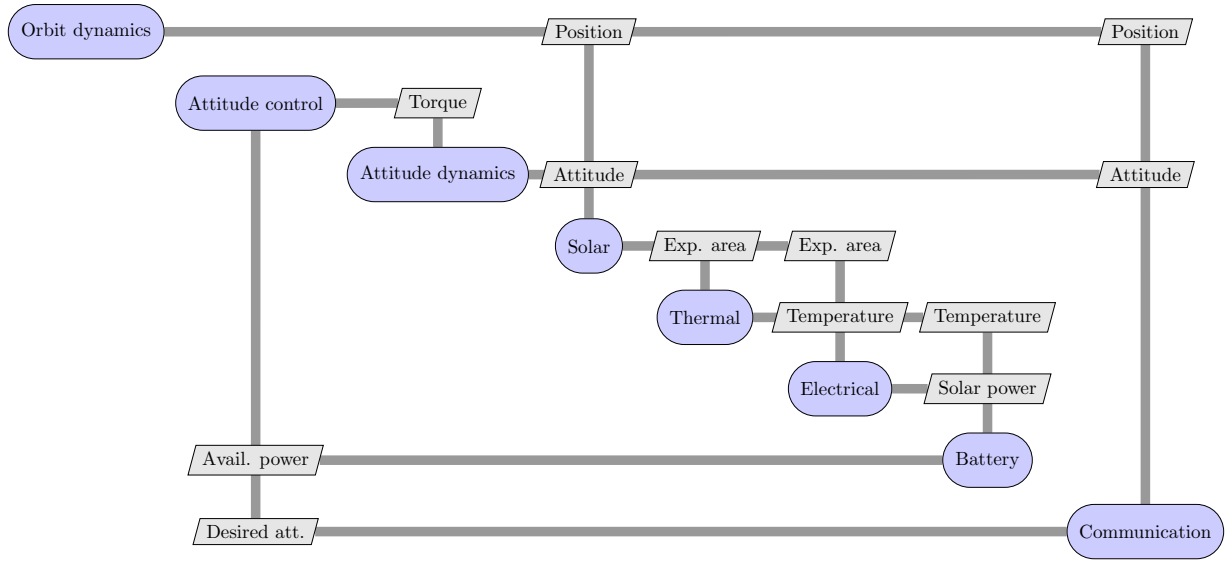
A. Optimization Problem

FORMULATION Figure 7 contains a variant of the eXtended Design Structure Matrix (XDSM) diagram [15] of the multidisciplinary system. The XDSM format is a visualization tool that simultaneously shows process and data flow for MDO processes. In this case, only data flow is relevant, shown by the thick grey lines. In Fig. 7, only the disciplines are listed as opposed to the full set of components for the sake of brevity, but the diagram shows the structure of the problem and the dependencies between components.

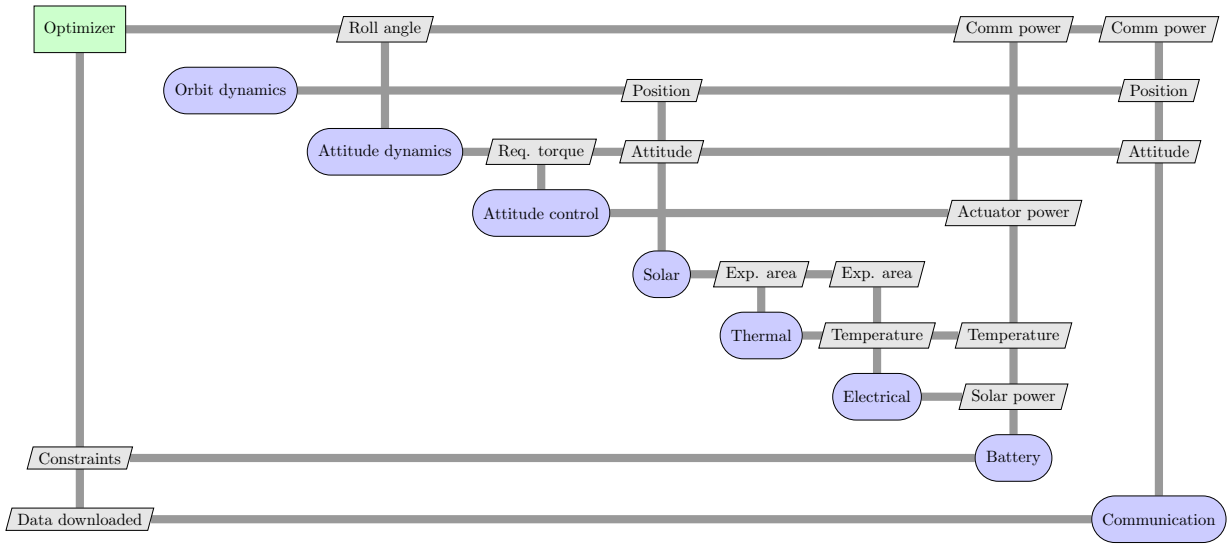
Fig. 7(a) shows that the true, physical problem formulated in a natural way has coupling at the discipline level in two feedback loops. The first is due to the use of a controller as the communication discipline computes data download rate, which depends on attitude, but it also passes a desired attitude variable back to the controller which reflects how

much the satellite must rotate to get an optimal link with the ground station. The second is more complex — attitude also affects the amount of solar power generated and heat accumulated, which affects battery performance and changes the amount of power the actuator has available to rotate the spacecraft.

By using optimization, the coupling is removed from the multidisciplinary system, as shown in Fig. 7(b). The optimizer is given full control over the satellite's attitude profile and battery charge constraints are explicitly handled by the optimizer, so the feedback loops present in Fig. 7(a) become the responsibility of the optimizer as opposed to the multidisciplinary analysis solver, which now sees a sequential problem. The optimizer also has control over the power allocated to the communication module for signal amplification, which gives it flexibility it can use to satisfy battery constraints and extract further gains in total data downloaded.



(a) True problem



(b) Decoupled problem

Figure 7: eXtended Design Structure Matrix (XDSM) diagrams [15] for the coupled and decoupled problems. Off-diagonal entires represent dependencies between components.

LARGE-SCALE OPTIMIZATION As previously mentioned, the full multidisciplinary problem involves multiple time scales. A single ground station pass occurs on the order of seconds, the satellite's orbit has a period on the order of minutes, and the earth's orbit around the sun and the precession of the satellite's orbit takes place over days and months. The solution is to take small time steps ($\mathcal{O}(10sec)$) to capture the first scale, use a long simulation period ($\mathcal{O}(10^4 sec)$) to capture the second scale, and run multi-point optimization to capture the third scale.

The approach is to run multiple simulations simultaneously, each one started at different times of the year. The third time scale is addressed by running high-resolution simulations from different months and averaging the results to approximately capture the overall performance of the spacecraft over a full year. Specifically, the optimization simulates 800 minutes of the spacecraft's operation at launch and at 3, 6, 9, and 12 months after launch. The data downloaded at the end of each of the five points are averaged to yield the objective function, and separate constraints are imposed for each point of the multi-point optimization.

As a result of the large number of time steps required along with multi-point optimization, the coupled multidisciplinary system we are considering involves 1,246,150 total variables that are solved for in each multidisciplinary analysis. Design variables that have values at each of the 1000 time instances are represented by B-splines with 500 control points in order to reduce the total number of design variables and also to ensure that the variables vary smoothly in time. With this reduction, the full multidisciplinary optimization problem has 35086 design variables in total.

OPTIMIZATION PROBLEM The objective of the optimization problem is to maximize total data downloaded, as the satellite's mission is to collect and relay to Earth as much data as possible. All other considerations such as solar power and battery are secondary as the functions of these subsystems are to enable the satellite to maximize data downloaded. There are two scalar design variables, fin angle and antenna angle, and 84 variables that represent whether a cell or radiator is installed. The profile variables — those for which we are optimizing the profile over time — are the power allotted to the communication discipline for amplification, roll angle, and 12 set point currents. The set point variable has the effect of emulating maximum power point tracking (MPPT) for the solar power module since the optimizer effectively picks the current, and indirectly, the voltage, at which the maximum power can be generated from the cells in a given solar panel. There are four inequality constraints at each of the five points, which are aggregated KS functions: two that limit the maximum charge (5 A) and discharge (10 A) rates, one that sets the maximum state of charge at 1, and another that sets the minimum state of charge at 0.2. The optimization problem is summarized below:

maximize	$\sum_{i=1}^{5} D_i$	data downloaded	
with respect to	$0 \leq I_{setpt} \leq 0.4$	Solar panel current	$500 \times 12 \times 5$
	$0 \leq \gamma \leq \pi/2$	Roll angle profile	500×5
	$0 \leq P_{comm} \leq 5$	Comm. power	500×5
	$0 \leq cellInstd \leq 1$	Cell vs. radiator	84
	$0 \leq finAngle \leq \pi/2$	Fin angle	1
	$0 \leq antAngle \leq \pi$	Antenna angle	1
		Total	35086
subject to	$I_{bat} \leq I_{max}$	Batt. charge con.	5
	$I_{min} \leq I_{bat}$	Batt. discharge con.	5
	$0.2 \leq SOC$	Batt. capacity con.	5
	$SOC \leq 1$	Batt. capacity con.	5
		Total	20

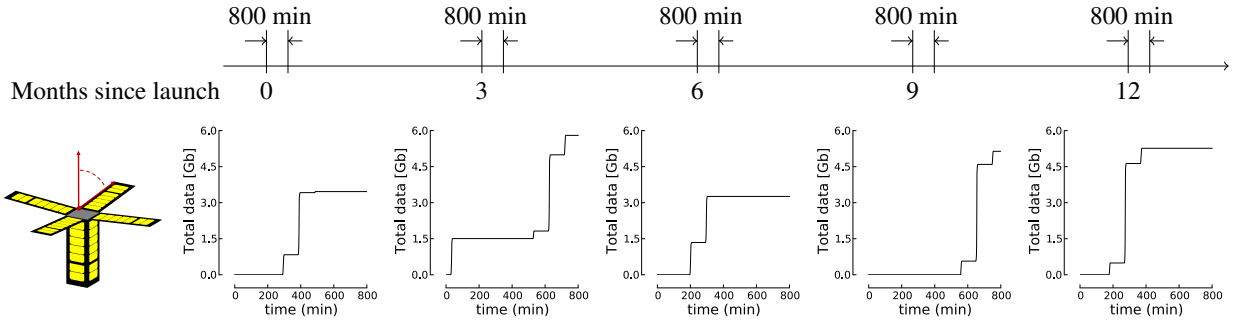
B. Optimization Results

The optimization problem was solved using SNOPT [16], a reduced-Hessian active-set sequential quadratic programming (SQP) optimizer that solves nonlinear constrained problems very efficiently, particularly when derivatives are provided as is the case here. Three optimizations were run, each corresponding to a different launch date and launch orbit. Since the target launch window is summer 2014, the three potential launches considered are April 1, July 1, and October 1, 2014.

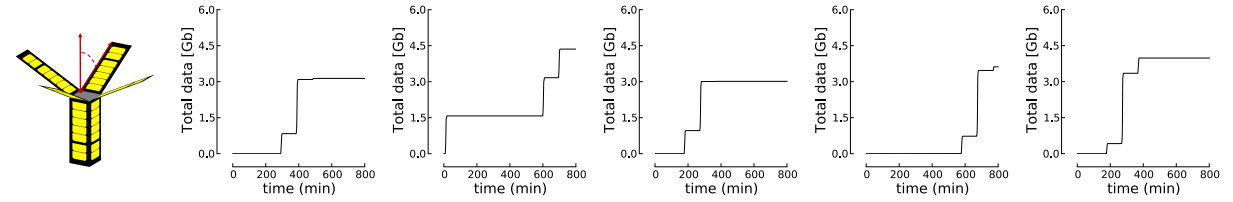
For all three optimizations, the initial design point was infeasible because the state of charge becomes negative due to insufficient power generation, but the choices of initial values for the design variables were reasonable. For the April 1 and July 1 optimizations, the optimizer was able to satisfy all constraints by the 31st and 16th major iterations, respectively; however, the October 1 optimization did not find a feasible point. Specifically, for three of the points, the optimizer could not generate sufficient power to maintain a battery state of charge greater than zero for the

October 1 optimization. The optimizations required, on average, roughly 10 hours to achieve a reduction of 2 orders of magnitude in optimality. There is no guarantee that the optimum found is the global optimum, but optimization improved the design without question.

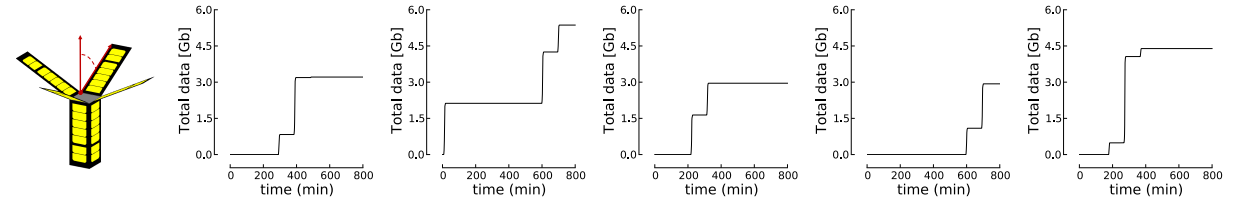
Figure 8 shows data downloaded plots for the five points for each of the three launch candidates. The results show that the April 1 launch yields the highest potential data downloaded as the average of the total data after 800 min over the five points is the greatest. Another interesting resulting result is that for the April 1 launch, a fin angle of $85.6°$ is optimal, whereas the optimal fin angle for the July 1 launch is $62.4°$. This contrasts with $67°$, which is the value found in single-discipline, single-variable optimization with power generated as the objective function. Also, removing the multi-point aspect of the optimization problem presented here yields an optimal fin angle of $83°$. It is evident that fin angle, which has a significant effect on power generation capability, is a highly sensitive variable and necessitates all aspects of the full MDO problem to be considered.



(a) April 1, 2014 launch: (optimal fin angle of $85.6°$)



(b) July 1, 2014 launch: (optimal fin angle of $62.4°$)

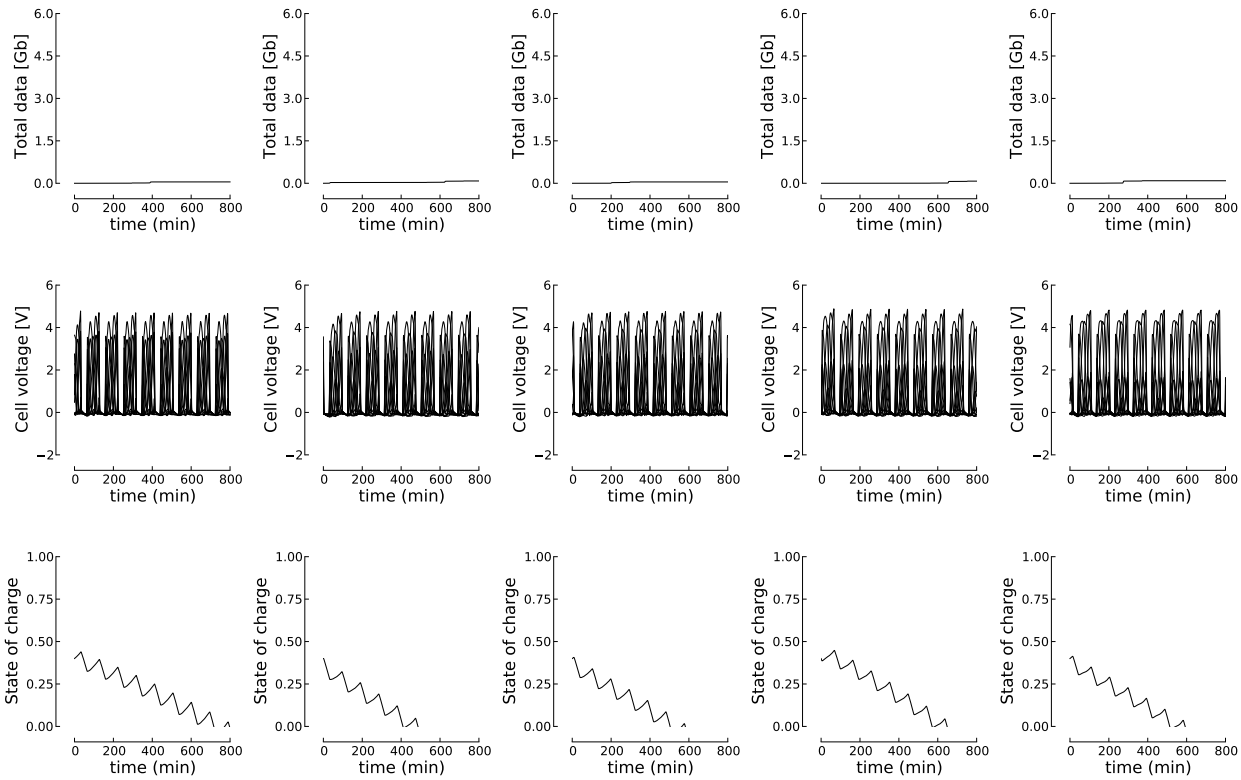


(c) October 1, 2014 launch: (optimal fin angle of $60.2°$)

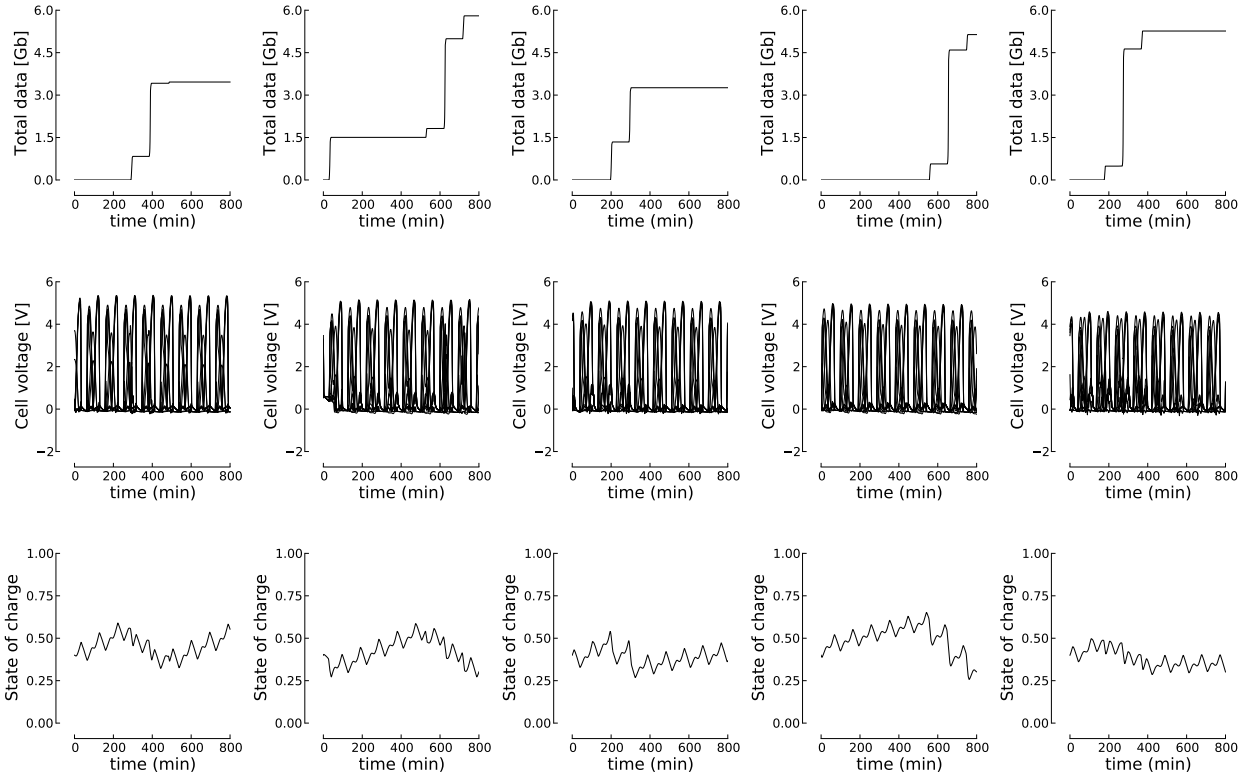
Figure 8: Results from three optimizations, corresponding to three different launch dates and launch orbits. The April 1 launch yields the highest data downloaded.

Figure 9 compares the initial and final values for total data downloaded, cell voltage, and state of charge for the April 1 optimization. The first difference to notice between the initial and optimized profiles is the significant increase in data downloaded. This difference is somewhat exaggerated by the fact that the initial signal amplification power is very low, so the data download rate has much room to increase during optimization. Moreover, the solar cell voltages for the 12 solar panels are all increased through optimization, which helps increase the battery state of charge. The state of charge curve is initially infeasible for all five points, but is made feasible through increased power generation.

Figure 10 examines the fifth point of the April 1 optimization in more detail. The first row — roll angle, set point current, and power allocated for communication — represents curves that the optimizer has control over as design variables, so Figure 10 shows how the optimizer was able to improve the design and ensure feasibility. In the roll angle plot, the changes in roll angle coincide with ground station passes (when the comm. line of sight is 1), so it is



(a) Initial



(b) Optimized

Figure 9: Results from the April 1 optimization.

clear the optimizer rotates the spacecraft to increase transmission gain, which can be seen by comparing the initial and optimized gain plots. The set point currents in the solar cells and the fin angle are used to increase the cell voltage, hence, the total solar power generated. As discussed previously, the additional solar power is used to ensure the battery state of charge always stays in the feasible region — between 20% and 100% — at the optimized design. Similar to roll angle, power allocated to communication also has temporary increases that coincide with ground station passes, as can be expected, and the peaks are limited by the design variable bound of 5 W.

C. Conclusion

The objective of this project was to apply large-scale multidisciplinary optimization to the design of a small satellite. To enable this, we developed a framework which greatly simplifies the task of defining a problem with many disciplines, automatically solves the multidisciplinary analysis, automatically computes coupled derivatives, and provides a lightweight platform for implementing a large variety of solution techniques. The ease-of-use and simplicity afforded by this framework enabled all the disciplines to be implemented as 43 distinct components (see Fig. 11) in less than a month. Other key enabling tools included multi-point optimization, constraint aggregation, smooth multi-dimensional interpolation, a differentiated Runge-Kutta ODE solver, and efficient numerical linear algebra. The result was that we were able to solve an MDO problem with 8 disciplines, more than 35,000 design variables, and over 1.2 million total variables that represent 800 minutes of the satellite's operation at five uniformly spaced points over the year.

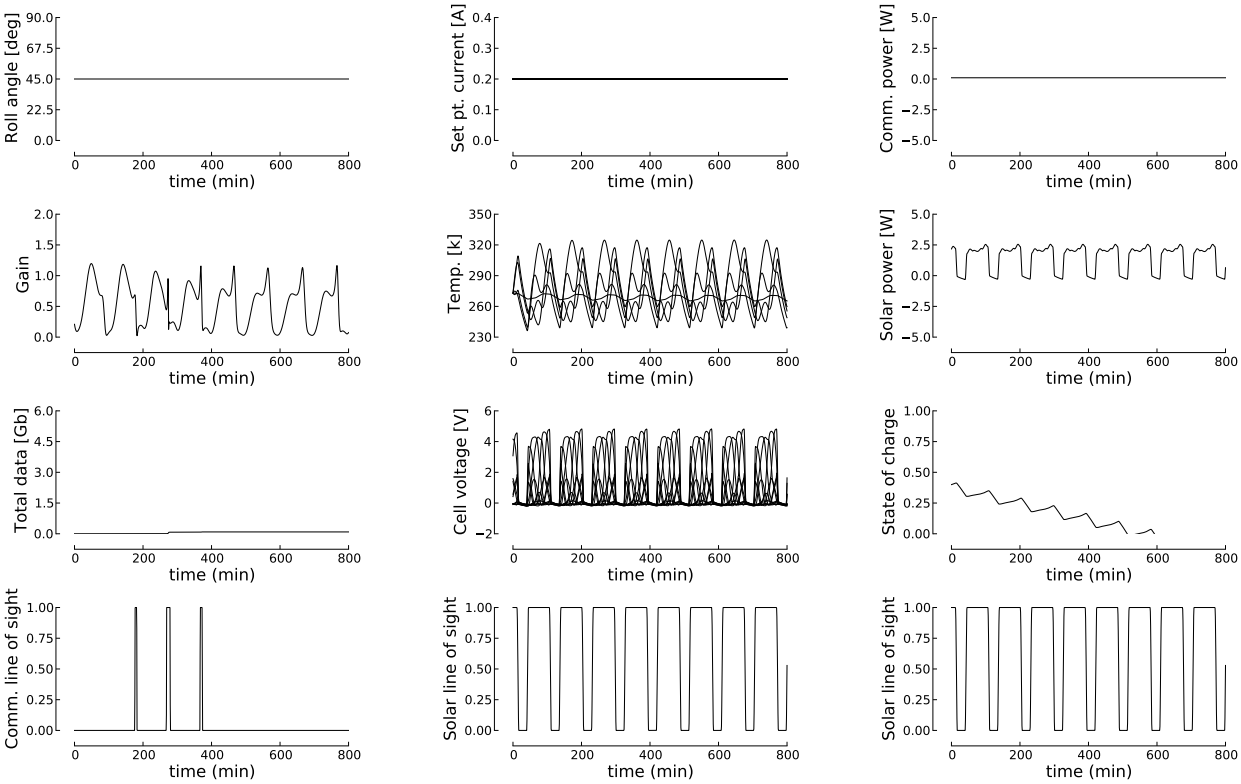
Through this work, it has become clear that combining small satellite design and MDO is beneficial to both fields. For the former, the large number of disciplines involved and the strength of couplings make it challenging to optimize or even analyze the coupled system in a monolithic fashion, but they are also why such an approach is needed. For the MDO field, the small satellite design problem is a good benchmarking problem and potentially an application where MDO algorithms and methods can make an impact. There are many disciplines, variables, and tradeoffs, and the fact that much of the behavior and coupling in the system can be captured with relatively inexpensive numerical models provides flexibility for design optimization.

There are a number of clear avenues for future work. The most obvious is to improve upon the models used in this problem and further explore the CADRE design problem with more fidelity. In terms of memory and computation costs, there is room to make the problem larger in order to account for more aspects of the problem or a longer satellite operation time.

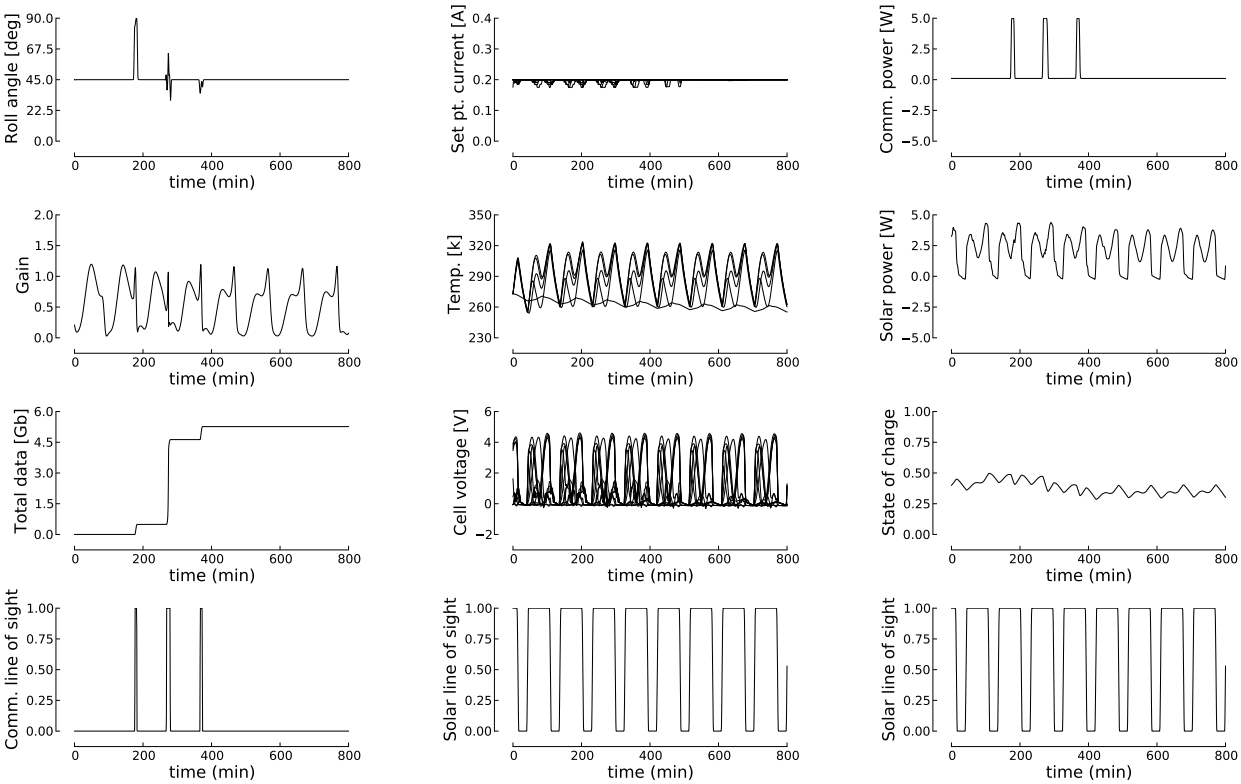
The MDO framework presented in this paper is also promising and has many obvious areas for further development. The first is to make it compatible with parallel components that store data on multiple processors. A solution for this was discussed in an earlier section whereby the system-level solver not only operates in a matrix-free fashion, but also 'vector-free'. Another step is to incorporate it in an existing, established framework such as OpenMDAO [17], which is currently being explored. Long-term, the hope is that this framework will make gradient-based optimization and other efficient solution techniques more accessible for general computational design problems.

VI. Acknowledgments

This work was partially supported by NASA through award No. NNX11AI19A. The authors would like to thank Daniel Meinzer, who developed the OpenGL-based exposed area model, and John Springmann for providing his knowledge of small satellites.



(a) Initial



(b) Optimized

Figure 10: Results from the fifth point (12 months after launch) of the April 1 optimization.

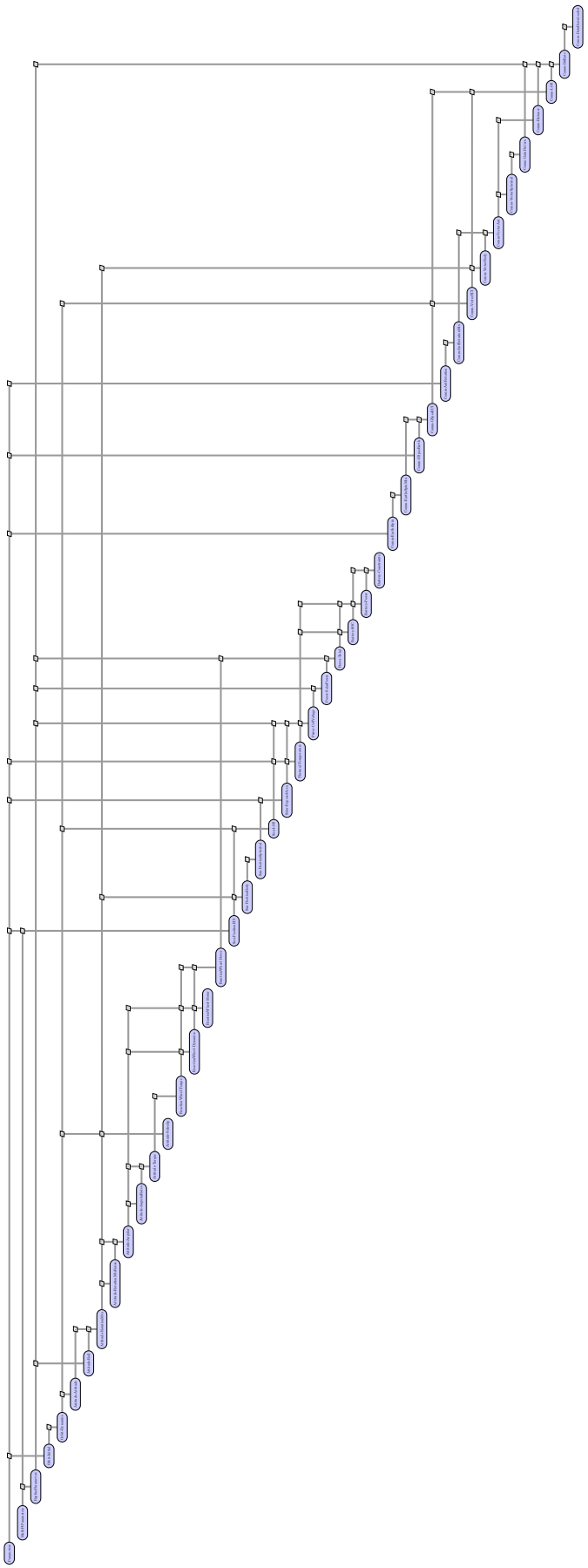


Figure 11: XDSM diagram of all the components implemented in the MDO framework.

References

[1] Cutler, J. W., Ridley, A., and Nicholas, A., "Cubesat Investigating Atmospheric Density Response to Extreme Driving (CADRE)," *Proceedings of the 25th Small Satellite Conference*, Logan, UT, August 2011.

[2] Ekpo, S. and George, D., "A system-based design methodology and architecture for highly adaptive small satellites," *4th Annual IEEE Systems Conference*, San Diego, CA, April 2010.

[3] Cvetkovic, S. R. and Robertson, G. J., "Spacecraft design considerations for small satellite remote sensing," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 29, No. 2, April 1993, pp. 391–403.

[4] Ashby, G. F. and Kramer, S. C., "Integrated design, evaluation, and decision making for small satellite design," *IEEE International Conference on Simulation*, Orlando, FA, October 1997.

[5] Sun, Z., Xu, G., Lin, X., and Cao, X., "The integrated system for design, analysis, system simulation and evaluation of the small satellite," *Advances in Engineering Software*, Vol. 31, No. 7, July 2000, pp. 437–443.

[6] Hoag, L., Kichkaylo, T., and Barnhart, D., "A systems architecting approach to automation and optimization of satellite design in SPIDR," *International Conference on Engineering and Meta-Engineering (ICEME)*, Orlando, FA, 2010.

[7] Spangelo, S. and Cutler, J., "Integrated Approach to Optimizing Small Spacecraft Vehicles and Operations," *Proceedings of the 62nd International Astronautical Congress*, Cape Town, South Africa, October 2011.

[8] Zingg, D. W., Nemec, M., and Pulliam, T. H., "A Comparative Evaluation of Genetic and Gradient-Based Algorithms Applied to Aerodynamic Optimization," 2007.

[9] Martins, J. R. R. A. and Lambe, A. B., "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA Journal*, 2013, (In press).

[10] Martins, J. R. R. A. and Hwang, J. T., "Review and Unification of Discrete Methods for Computing Derivatives of Single- and Multi-disciplinary Computational Models," *AIAA Journal (Accepted subject to revisions)*, 2013.

[11] de Boor, C., *A Practical Guide to Splines*, Springer, 2001.

[12] Giesselmann, J., "Development of an Active Magnetic Attitude Determination and Control System for Picosatellites on highly inclined circular Low Earth Orbits," *Master of Engineering RMIT University*, January 2006, pp. 1–11.

[13] Kawamura, H., Naka, K., Yonekura, N., Yamanaka, S., Kawamura, H., Ohno, H., and Naito, K., "Simulation of I-V characteristics of a PV module with shaded PV cells," *Solar Energy Materials & Solar Cells*, Vol. 75, 2003, pp. 613–621.

[14] Larson, W. and Wertz, J., *Space mission analysis and design*, Kluwer Academic Publishers, January 1991.

[15] Lambe, A. B. and Martins, J. R. R. A., "Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes," *Structural and Multidisciplinary Optimization*, Vol. 46, 2012, pp. 273–284. doi:10.1007/s00158-012-0763-y.

[16] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal of Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006. doi:10.1137/S1052623499350013.

[17] Heath, C. and Gray, J., "OpenMDAO: Framework for Flexible Multidisciplinary Design, Analysis and Optimization Methods," *Proceedings of the 53rd AIAA Structures, Structural Dynamics and Materials Conference*, Honolulu, HI, April 2012, AIAA-2012-1673.