Efficient and Scalable Computational Design of a Small Satellite

John T. Hwang
Department of Aerospace Engineering, University of Michigan
1320 Beal Ave, Ann Arbor, MI 48109; 734-764-3310
hwangjt@umich.edu

Faculty Advisor: Prof. Joaquim R. R. A. Martins
Department of Aerospace Engineering, University of Michigan

ABSTRACT

The design of a power-constrained CubeSat is a complex problem involving several disciplines that are coupled. Algorithms and hardware for scientific computing have advanced to the point that recently, design and operations optimization of a CubeSat was successfully performed involving 7 disciplines, over 25,000 design variables, and roughly 2.2 million total variables modeled. This paper addresses the bottlenecks of this algorithm in computer memory costs and execution time. This is done through a new parallel computational modeling framework that automates many aspects of distributed-memory parallel computing, and an analytic approximation for the solar cell model that is significantly more efficient than the previous model. Results show improved scaling of execution time with the number of unknowns in the problem and nearly an order-of-magnitude improvement in gradient computation time.

I. Introduction

Small satellites offer a useful, cost-effective platform for performing research and testing new technologies. Their compact scale attracts both academia and industry because of the significantly lower cost to build and launch compared to larger satellites. The development time can also be much shorter, enabling some teams to develop and launch on the order of one satellite per year.

CADRE, from "CubeSat investigating atmospheric density response to extreme driving", is funded by the National Science Foundation with the objective of studying the Earth's ionosphere and thermosphere [4]. Motivated by the rapid growth of space-based infrastructure, CADRE will collect space weather data in low-Earth orbit which will be used to help understand how the upper atmosphere responds to energy inputs. CADRE is a power-constrained design due to the persistent power demands of the data-collection instruments and the need to transmit large amounts of data to ground stations on Earth.

The design of CADRE (shown in Fig. 1) is a challenging problem because of the complexity and the number of disciplines that must be considered. Its ability to generate and store sufficient power requires effective management of its temperature since extreme heat or cold can adversely affect solar panel and battery performance. CADRE's reaction wheels can be used to alter its orientation to shade areas of the satellite when it is overheated, but the power usage of the actuators and the potential for reducing Sun exposure must be simultaneously considered. Clearly, the design of CADRE involves coupling that must be considered when choosing geometric variables to be able to predict how it will operate.

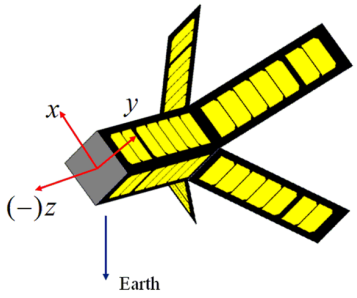

Figure 1. Rendering of the CADRE CubeSat.

Given the short development cycle of small satellites, computational tools can play an important role by enabling rapid exploration of potential designs. Optimization in particular can be very useful to automate what might otherwise be a manual and repetitive pro-

cess of performing analyses on potential designs, especially when large numbers of design variables are involved. For this reason, there has been much work in the literature in which computational modeling and optimization have been used in the design of satellites. For instance, researchers have explored the use of optimization on the structure [3], thermal control system [7], battery load scheduling [10], attitude control system [15], and layout [17].

Other researchers have taken a holistic approach by optimizing multiple disciplines simultaneously [2, 6, 14, 16, 5, 9]. In all of these cases, a gradient-free optimization algorithm is used with the common approach of intelligently selecting a large number of designs to analyze, but this approach does not scale efficiently since evaluating only 2 values for each design variable already results in 2^n evaluations with n being the number of design variables. The large number of required evaluations must be balanced by limiting the scope and detail of the analyses to ensure each evaluation is very fast or by limiting the number of design variables.

For the design of CADRE, there are three aspects of the problem that necessitate an approach without these limitations. First, it is a power-driven design, but to adequately capture the effects of the design variables that affect CADRE's power generation and storage, several more disciplines are drawn into the problem, including attitude control, communication efficiency, battery performance, temperature, and solar power generation. Second, modeling CADRE involves multiple time scales because its ground station passes only last $\mathcal{O}(\text{s})$, its orbit and power-related quantities are periodic on the scale of $\mathcal{O}(\text{min})$, and its orbit precession period is $\mathcal{O}(\text{months})$. It is challenging to maintain sufficient accuracy on the smallest scale and to span the full extent of the largest scale while maintaining efficiency. Finally, it would be preferable to model or optimize the operational characteristics of the satellite simultaneously with the design because they affect each other.

The cost and complexity of both implementation and execution increase significantly when trying to solving the true, large-scale problem without compromising on these three points. To mitigate these challenges, very recent work done by the author and collaborators used a gradient-based approach with adjoint-based derivative computation [8]. The combination of a quasi-Newton sequential quadratic programming (SQP) optimizer and the adjoint method significantly reduces the rate of increase of execution time versus the number of design variables, enabling optimization with over 25,000 design variables in this case. The second enabling factor was the use of a computational modeling framework to simplify the implementation of the problem involving 7 disciplines. The framework takes a centralized approach to the implementation whereby each block of code is written independently and the framework automates passing data between codes, converging the global problem, and computing coupled derivatives using a unified equation [13].

This algorithm was successful in solving the intended problem, but there are many avenues for future work to fully maximize the potential of the approach. Due to the modular implementation, it is extensible to constellations of satellites, multiple ground stations, additional geometric design variables, and more resolution for higher accuracy. However, the algorithm was memory-limited as it approached the memory available on a desktop computer. Furthermore, it required 100 hours to solve the optimization problem, limiting its utility in an actual design setting because of the long turnaround time.

This paper presents a significantly more scalable and efficient method for solving the large-scale optimization problem presented in Hwang et al. [8] It does this through the development of a general framework that enables parallel computing and, further, automates distributed memory parallelism so that the memory limits are overcome by splitting data across computing nodes. The new framework also hierarchically decomposes the multidisciplinary problem, allowing it to solve the same problem in a more modular, extensible, and efficient way.

The paper will describe the theory underlying the framework, highlight key algorithmic details, summarize the models for the disciplines, and present results.

II. Theory

This section will review the theory underlying the parallel computational modeling framework. Much of the theory overlaps with the earlier framework presented in Hwang et al. [8]. The differences are for the most part in the implementation and software architecture, which will be presented in Sec. III.

A. Formulation

The fundamental idea behind the theory of the framework is the formulation of the computational modeling problem as a system of algebraic equations. Depending on the application, computational models work with many types of variables with many different names — e.g. input, output, state, behavioral, design, parameter, coupling, objective, constraint, and intermediate. Conceptually, this forces any general treatment or unification of computational models to be heterogeneous with specific cases for handling each type of variable. This is the motivation for concatenating all variables into a single vector without discriminating their type or origin, and this forms the vector of unknowns for the system of equations.

The next step is to define a function for each variable

that constrains it to the right value. When these functions are also combined to form a single vector-valued function, the result is the nonlinear algebraic system given by

$$\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{0} \quad , \quad \boldsymbol{F} : D \subseteq \mathbb{R}^n \to \mathbb{R}^n \tag{1}$$

where boldface is used to denote vectors and vector-valued functions. To be explicit, the unknown vector $\boldsymbol{u}$ is partitioned into the constituent variables $\boldsymbol{v}^{(i)}$:

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{v}^{(1)} \\ \vdots \\ \boldsymbol{v}^{(N)} \end{bmatrix} \quad \boldsymbol{u} \in \mathbb{R}^n \ , \ \boldsymbol{v}^{(i)} \in \mathbb{R}^{n_i} \ 1 \leq i \leq N \tag{2}$$

The remaining question is how to define $\boldsymbol{F}^{(i)}$ appropriately for a given variable. Any variable can be classified as one of three types: independent, where its value is set outside of the algebraic system; explicit, where its value is an explicit function of other variables; or implicit, where its value is constrained by a residual function that must be driven to zero. If $\boldsymbol{v}^{(i)}$ an independent variable set to the value $\boldsymbol{a}$, the choice would be $\boldsymbol{F}^{(i)}(\boldsymbol{v}) = \boldsymbol{v}^{(i)} - \boldsymbol{a}$ so that when $\boldsymbol{v}^{(i)} = \boldsymbol{a}$, it follows that $\boldsymbol{F}^{(i)}(\boldsymbol{v}) = \boldsymbol{0}$, as desired. If it is an explicit variable defined by $\boldsymbol{v}^{(i)} = \boldsymbol{G}^{(i)}(\boldsymbol{v}^{(j \neq i)})$, the natural choice would be $\boldsymbol{F}^{(i)}(\boldsymbol{v}) = \boldsymbol{v}^{(i)} - \boldsymbol{G}^{(i)}(\boldsymbol{v}^{(j \neq i)})$. In the implicit case, the choice $\boldsymbol{F}^{(i)}$ for a variable $\boldsymbol{v}^{(i)}$ is clear; it should just be equal to the residual function.

B. Significance

The result of formulating the general computational modeling problem as an algebraic system is that the process of running a simulation reduces to the task of solving this algebraic system. Furthermore, it greatly simplifies the implementation of any framework built on this theoretical foundation because it can internally store and manage the problem as a homogeneous algebraic system, and the distinction between whether a particular variable is an input, output, parameter, state, etc. is only manifested in computing $\boldsymbol{F}^{(i)}(\boldsymbol{v})$ for a particular variable. The remaining tasks of the framework, including passing data between codes and solving the coupled system, can be ignorant of the type of each variable.

C. Derivatives

A key implication of this formulation is the ability to compute coupled derivatives in a simple manner. If $\partial \boldsymbol{F} / \partial \boldsymbol{u}$ is invertible, there exists a local inverse $\boldsymbol{F}^{-1}$ defined on an open neighborhood of the point in the codomain. Moreover, the Jacobian of the inverse is the inverse of the Jacobian, leading to the equation presented in Martins and Hwang [8]:

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}} \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{f}} = \mathcal{I} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}^T \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{f}}^T \tag{3}$$

The significance of this equation is that by specifying only the Jacobians of partial derivatives $\partial \boldsymbol{F}^{(i)} / \partial \boldsymbol{u}$ for each variable $\boldsymbol{v}^{(i)}$, it is possible to compute the total derivative of any variable i with respect to variable j, i.e. $d\boldsymbol{v}^{(i)} / d\boldsymbol{v}^{(j)}$. Furthermore, a row or column of the Jacobian of total derivatives is simultaneously computed at the cost of solving a linear system in Eq. (3). Thus, it is possible to efficiently compute the derivatives of the objective or constraints of an optimization with respect to an arbitrary number of design variables using the right-hand equality of Eq. (3), which is equivalent to the adjoint method. This is what enables efficient gradient-based optimization because a *single* solution of an $n \times n$ linear system yields derivatives that would otherwise require $n + 1$ solutions of an $n \times n$ nonlinear system using finite differences.

III. Software Architecture

This section describes the implementation of the parallel computational modeling framework, with particular emphasis on the hierarchical and distributed computing aspects.

A. Overview

At a high level, the parallel computational modeling framework is a tool that facilitates the implementation and execution of computational models. It enables the user to write codes independently of other codes by taking a modular overall approach and enforcing that each unit of code conforms to a small application programming interface (API). Fundamentally, its purpose is to run computational models involving multiple codes and to compute their derivatives, but in doing this it also provides built-in solvers, automates data passing, and provides useful utilities such as automatic validation of derivatives. The framework is implemented in Python and uses many solvers and parallel communication tools from the software package PETSc[1] via petsc4py. It is designed to have minimal overhead for both small- and large-scale codes, and is useful for optimization and analysis.

B. Problem Decomposition

Using ideas from Sec. II, the framework represents the computational model as a set of variables then defines and solves systems of algebraic equations on those variables. The framework uses a hierarchy of systems where systems contain other systems and each one is defined by the

variables it owns. It is useful to think of a hierarchy tree in which the leaves of the graph are the lowest-level systems which directly own the variables. All other nodes have children through which their variables are implicitly defined.

Figure 2 shows the hierarchy tree for a sample problem for illustration. The rectangles in green are called elementary systems since they do not contain other systems, while those in blue are called compound systems. Compound systems can be either serial or parallel, which refers to how the processors in a parallel computation are distributed. In the Message Passing Interface (MPI) standard, each system is given an MPI communicator, which defines the group of processes on which the system's operations are performed, so a serial system passes all of its processors to its children while a parallel system partitions its group of processors among the systems that are its children.

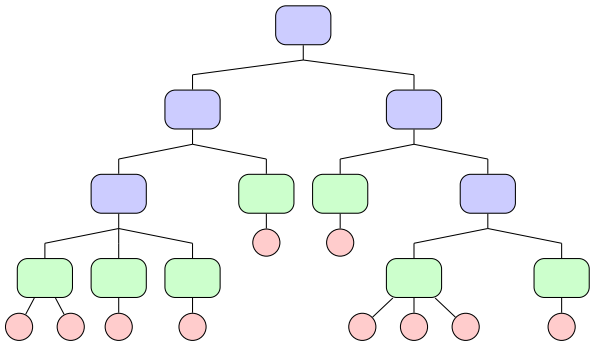

Figure 2. Hierarchy tree for a notional problem. The circles are variables and the rounded rectangles are systems with those in green considered the leaves of the tree (if the circle nodes are ignored).

C. Execution

Each unit of code inherits from a base *System* class and implements a small number of methods with which the framework controls execution order and calls the right methods in the right sequence to achieve convergence. The inherited classes implement six methods:

- Initialize: declares variable size and other properties; declares arguments and indices thereof that are needed
- Linearize: optionally assembles the Jacobian matrix and pre-computes factorizations if applicable
- Apply $\boldsymbol{F}^{(i)}$: compound systems simply delegate by calling this method in each of its child systems
- Apply $\frac{\partial \boldsymbol{F}^{(i)}}{\partial \boldsymbol{v}}$: compound systems simply delegate by calling this method in each of its child systems
- Solve $\boldsymbol{F}^{(i)}(\boldsymbol{v}) = \boldsymbol{0}$ for $\boldsymbol{v}^{(i)}$: the framework provides Newton, Gauss-Siedel, and Jacobi solvers or the user could implement their own solver
- Solve $\frac{\partial \boldsymbol{F}^{(i)}}{\partial \boldsymbol{v}^{(i)}}\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{x}$: the framework provides Krylov subspace (PETSc), Gauss-Siedel, and Jacobi solvers or the user could implement their own solver, which could be a preconditioner

A key feature is that all Jacobians and matrices are requested by the framework as a matrix-vector product operation, and the full matrix is never provided. This greatly simplifies the method because the framework does not make any assumptions whether the user implements a sparse, dense, factorized, or matrix-free Jacobian as all that is required by the framework is the effect of multiplying the matrix with a given vector.

Method	System	Elementary	Compound
initialize		User	Recursion
linearize		User	Recursion
apply_F		User	Recursion
apply_dFdpu	FD	Optional	Recursion
solve_F	Newton	Optional	GS/Jacobi
solve_dFdu	KSP	Optional	GS/Jacobi

Table 1. Implementation of the API for each type of system.

The framework can be described as taking a centralized approach because after initialization, a system is not aware of any information outside of itself. It provides data buffers for variables $\boldsymbol{u}$, arguments $\boldsymbol{p}$, and function values $\boldsymbol{f}$, as well corresponding vectors for the linear problem. The framework fills the input buffers with appropriate data, calls the right method from the system, and processes the data in the output buffers, so the local system has no knowledge regarding where the input data came from and how the output was processed. In the case of $\boldsymbol{p}$, the framework automatically fetches the data, which may be located on a different processor, by performing the necessary parallel data communication. Figure 3 illustrates an isolated system and shows how the data buffers form the interface between the system and the rest of the framework.

The data storage and accessing model is an aspect of the framework's design that is critical to its efficiency. Many global operations in the framework and its solvers are significantly more efficient when performed on large, contiguous vectors rather than smaller vectors that are scattered by variable. These operations could slow down significantly and unnecessarily when looping over all variables, particularly when there is a large number of variables. As shown in Fig. 8, the framework avoids this potential issue by storing all data as large, contiguous vec-

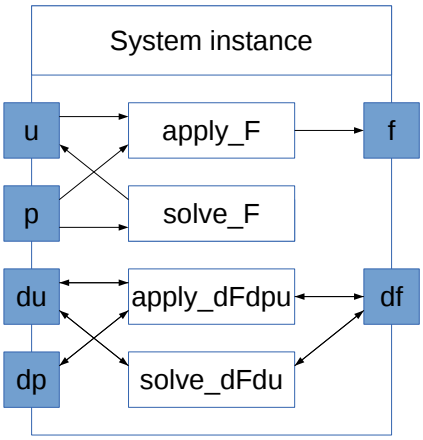

Figure 3. Data buffers and the corresponding operations for an isolated system. The double-headed arrows show that because there are two modes: forward and reverse.

tors and keeping a dictionary of views onto subvectors of the larger vector. The NumPy package provides array objects that do not own unique data, but rather, contain pointer, scope, and stride information, enabling fast vector operations on the sub-vectors with the speed of a compiled language. Furthermore, the user can access data intuitively using strings as keys for a variable that is part of a larger array, without knowing the global indices.

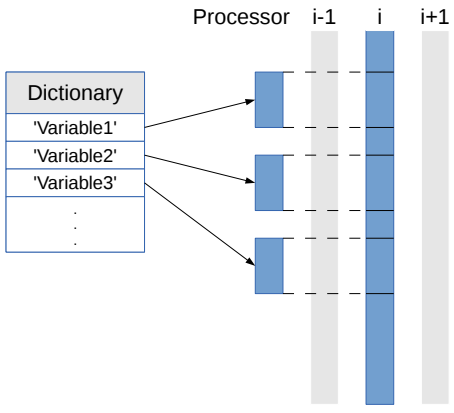

Figure 4. Large, contiguous vectors are stored, but accessed as a dictionary of views of subvectors.

D. CADRE Implementation

The parallel computational modeling framework has been designed in a way such that it can be useful for a broad range of fields, problems, and algorithms. The implementation of the CADRE problem, however, makes use of a subset of the framework's capabilities and has several unique features.

The first unique characteristic is the scope and number of variables in the CADRE design problem. As Fig. 5 shows, there are a large number of disciplines, each of which further subdivide to yield 63 units of code in total with a complex network of dependencies. The framework facilitates management of these dependencies by enabling implementation of each code in sequence and a tool for automatically checking partial derivatives.

Another characteristic is multi-point analysis and optimization. The multi-scale nature of the design problem is addressed by performing multiple 12 hour simulations spread out over the year, and the framework automates the process of making multiple instances of simulations and gathering output data from them. Furthermore, the framework automates parallel execution by enabling the points to be distributed across available processors and all operations involving scattering and gathering data across processors is inherently performed by the framework. This allows parallel computing with distributed data without writing a single line of parallel code from the user's perspective.

IV. Discipline Models

This section lists and briefly summarizes the modeled disciplines. The reader is referred to [8] for more detailed information on the models with the exception of solar cell voltage, for which a new model has been developed.

A. Orbit Dynamics

The orbit-dynamics discipline involves numerical integration of the equations of motion with the J perturbation terms included to capture the precession of the satellite's orbit. A 4th-order Runge-Kutta scheme is used to integrate this and all other ordinary differential equations (ODEs) modeled in this work.

B. Attitude Dynamics

The scientific requirements of the mission constrain CADRE to have a forward-facing orientation at all times, though the roll angle is permitted to vary. The roll angle profile is controlled by the optimizer via a B-spline parametrization. In this work, 4th order B-splines are used to parametrize all profile design variables such as the roll angle, with the number of B-spline control points equal to roughly a quarter of the number of points in the discretization as a rule of thumb. Using B-splines to parametrize the profiles yields 2 benefits—it reduces the number of design variables and only permits smooth profiles in the

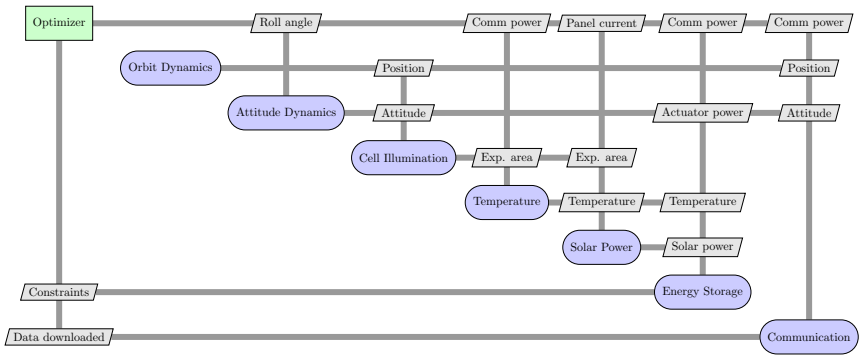

Figure 5. Extended Design Structure Matrix (XDSM) [12] diagram showing the coupling in the CADRE design problem [8].

design space. An analytical model has been developed to smoothly approximate manufacturer-provided data for the reaction wheels, whose inputs are computed based on the optimizer-specified attitude profiles.

C. Cell Illumination

The cell illumination discipline is important because it captures the projected normal area of sunlight to which each solar cell is exposed. A table of data has been generated by running OpenGL simulations in a discrete fashion to simplify capturing shading. Therefore, a multi-dimensional tensor-product B-spline was required to interpolate this table of data with a smooth function that also provides analytic partial derivatives. This discipline also models the line of sight between the satellite and the Sun.

D. Temperature

The satellite is assumed to have 5 regions of uniform temperature—the 4 fins and the body. The temperature ODE accounts for heat absorption from the Sun, constant radiation of heat, and heating from the communication subsystem.

E. Solar Power

The solar cell model is based on an implicit equation [11] for voltage as a function of current, temperature, and exposed cell area. In the previous work [8], the numerical issues associated with solving an implicit equation were avoided by pre-computing solutions of the implicit equation and fitting another instance of the multi-dimensional B-spline interpolant. However, this was a bottleneck in the execution of the overall computational model because of the cost of evaluating this interpolant tens of thousands of times. This bottleneck was removed by developing an explicit approximation in this work.

This approximation uses the hyperbolic tangent function and attempts to fit the implicit model by matching the short-circuit current, slope at that point, diode voltage, and the open-circuit voltage.

The existing model is given by

$$\begin{Bmatrix} I_{sc} - I_{sat}\left[\exp\left\{\frac{V}{V_T}\right\} - 1\right] - \frac{V}{R_{sh}} - I = 0, I <= I_{sc} \\ V(I) = V_0 \tanh\left[\frac{-V_T R_{sh}}{V_0(I_{sat}R_{sh}+V_T)}(I - I_{sc})\right], I > I_{sc} \end{Bmatrix} \tag{4}$$

All constants and variables are defined in [8]. To estimate the open-circuit voltage, the first observation was that in Fig. 6, there is a localized region of high curvature around roughly 0.7 V to 0.9 V in all cases. Implicit differentiation yields an expression for dV/dI that is of the form $1/x$, and setting $x = 1$ gives an approximation for the point of highest curvature. Dropping a term close to unity and adding a factor of 1.1 since the open-circuit voltage is slightly beyond this point, the result is

$$V_{oc} = 1.1 V_T \ln \frac{V_T}{I_{sat} R_1} \tag{5}$$

where $R_1 = 1$ if in SI units. Another condition is that the slope dV/dI should be equal to that of the original model at I_{sc}, which is given by

$$\frac{dV}{dI} = -\frac{V_T}{V_T + I_{sat} R_{sh}} R_{sh} \tag{6}$$

Incorporating the bounds of V_0 and V_{oc}, the explicit equation for V is

$$V(I) = \frac{V_{oc} + V_0}{2} + \frac{V_{oc} - V_0}{2} \tanh\left[b(I - I_{sc}) + \operatorname{arctanh}\left(\frac{V_0 + V_{oc}}{V_0 - V_{oc}}\right)\right] \tag{7}$$

The coefficient b can be chosen using to satisfy Eq. (6), but removing the factor of 2 that results turns out to produce

curves that are more accurate:

$$b = \frac{1}{V_{oc} - V_0} \frac{dV}{dI} \tag{8}$$

Figure 6 shows how the implicit and explicit models compare for three choices of temperature and cell illumination area.

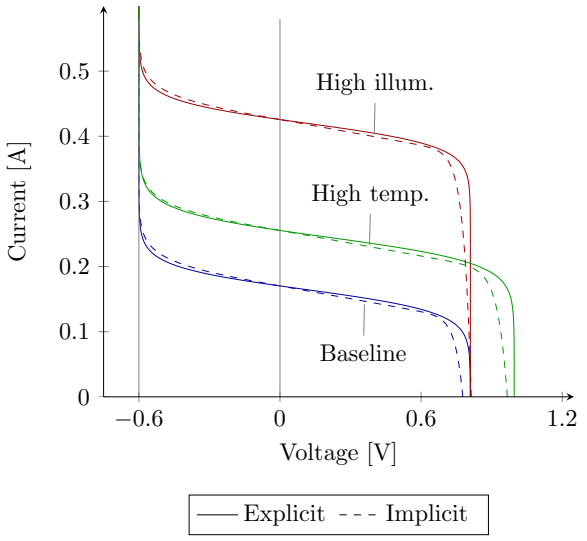

Figure 6. Comparison between the previous (explicit) and new (implicit) models for the solar cell I-V curve.

F. Energy storage

The energy-storage model integrates another ODE using the aforementioned RK4 solver. The battery's state of charge (SOC) is modeled considering the effect of temperature on battery performance. The terms that contribute to battery power drain include the reaction wheels, communication module, and a background 2 W power consumption by the scientific instruments and other subsystems.

G. Communication

For the communication discipline, a simple equation is used to model the relationship between bit rate, transmitter gain, signal-to-noise ratio (SNR), distance to the ground station, and power. A minimal acceptable SNR of 5.0 dB is assumed, which enables the computation of bit rate since all other quantities are computed in other models. The bit rate is integrated to compute the cumulative data downloaded profile, of which the final value is used to formulate the objective function.

V. Results

This section presents timings and derivative profiles computed using the implementation of the full CADRE simulation in the parallel framework.

The large-scale optimization of CADRE performed in the previous work was successful in achieving an 80 % improvement in the total data downloaded as shown in Fig. 7. However, the framework's efficient derivative computation capability also enables visualization of sensitivities such as in Fig. 8. The three figures on the left plot the derivatives of various quantities over time with respect to the fin angle variable, while the three figures on the right plot the derivatives of the minimum battery SOC constraint (20%) with respect to quantities over time.

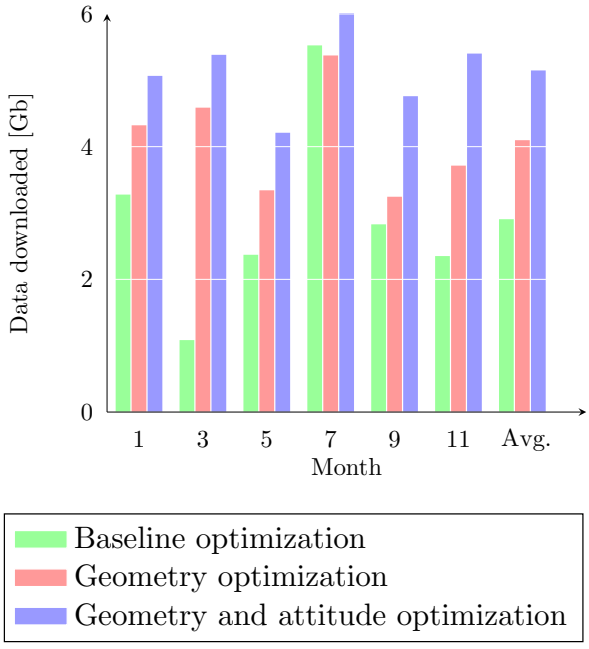

Figure 7. Division of total data downloaded over six simulations spread over the year for three optimization problems: a baseline optimization, geometric design variables added, and both geometric and operational variables added [8].

This plot shows the effectiveness of the direct and adjoint methods for computing derivatives. For the direct method the derivatives of all variables modeled in the system with respect to one variable are computed simultaneously at the cost of just a single solution of a linear system; likewise, for the adjoint method the derivatives of one output of interest with respect to all variables are computed simultaneously by solving a different linear system—the reader is referred to Martins and Hwang [13] for more details on the direct and adjoint methods. The efficiency of the framework enables a fast turnaround time for comput-

ing and plotting sensitivity information, which helps the designer interpret and augment large-scale optimization results and understand the design problem and tradeoffs more deeply.

As an example, the upper left plot in Fig. 8 would lead to the conclusion that increasing the fin angle at this particular design point would be beneficial. The average cell illumination would increase, leading to more solar power generation, which is also evident in the solar power plot, and the subsequent effect on the battery SOC is also shown in the lower left plot. In practice, there is more information that must be considered such as the simulations at other conditions—i.e. other points of the year—but an efficient and comprehensive modeling tool can be useful nonetheless.

Figure 9 presents timing results for the evaluating of the model and the computation of derivatives. The figure shows significant improvement in the new framework in evaluation time, but also noteworthy is how both scale with the number of unknowns. Since this is a log-log plot, the slope yields the order if there is a polynomial relationship between the two quantities, and the new framework shows a lower order, particularly for derivative computation. These results are promising as they imply that the gap is expected to increase rapidly as the size of the problem increases.

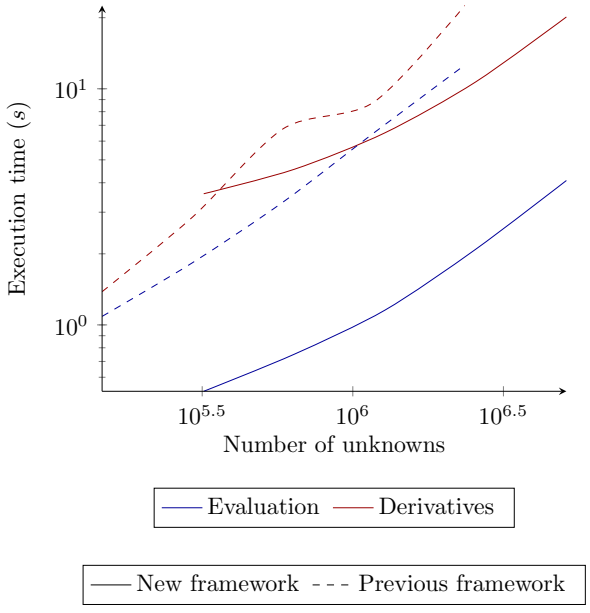

Figure 9. Improvement in efficiency of the new computational modeling framework, compared to the algorithm used in [8], both run on a single processor for comparison.

VI. Conclusion

This paper presented the extension of an algorithm for performing large-scale design and operations optimization of a small satellite, with the objective of improving scalability and facilitating future work. There were 3 main contributions. First, a parallel computational modeling framework was developed, which supports distributed memory parallel computing and improves upon the previous framework in efficiency and modularity. Second, a more efficient model was developed for the solar power discipline by deriving an explicit approximation for an implicit equation. Finally, it was shown that with the new framework, the algorithm can perform an evaluation nearly an order of magnitude faster and the derivative computation scales much better.

VII. Acknowledgments

The author would like to acknowledge Prof. Joaquim R. R. A. Martins, Dae Young Lee, and Prof. James W. Cutler, who contributed to the development of the CADRE MDO algorithm. The author would also like to thank Justin S. Gray, Kenneth T. Moore, Tristan A. Hearn, and Bret A. Naylor for insightful discussions. This work was partially supported by NASA through award No. NNX11AI19A–Technical Monitor: Justin S. Gray.

References

[1] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[2] David A Barnhart, Tatiana Kichkaylo, and Lucy Hoag. Spidr: Integrated systems engineering design-to-simulation software for satellite build. In *Proceedings of the 7th Annual Conference on Systems Engineering Research*, Loughborough, UK, 2009.

[3] A. Boudjemai, M. H. Bouanane, L. Merad, and A. M. Si Mohammed. Small satellite structural optimisation using genetic algorithm approach. In *Proceedings of the 3rd International Conference on Recent Advances in Space Technologies*, pages 398–406, Istanbul, Turkey, 2007.

[4] James W. Cutler, Aaron Ridley, and Andrew Nicholas. Cubesat investigating atmospheric density response to extreme driving (cadre). In *Proceedings of the 25th Small Satellite Conference*, Logan, UT, August 2011.

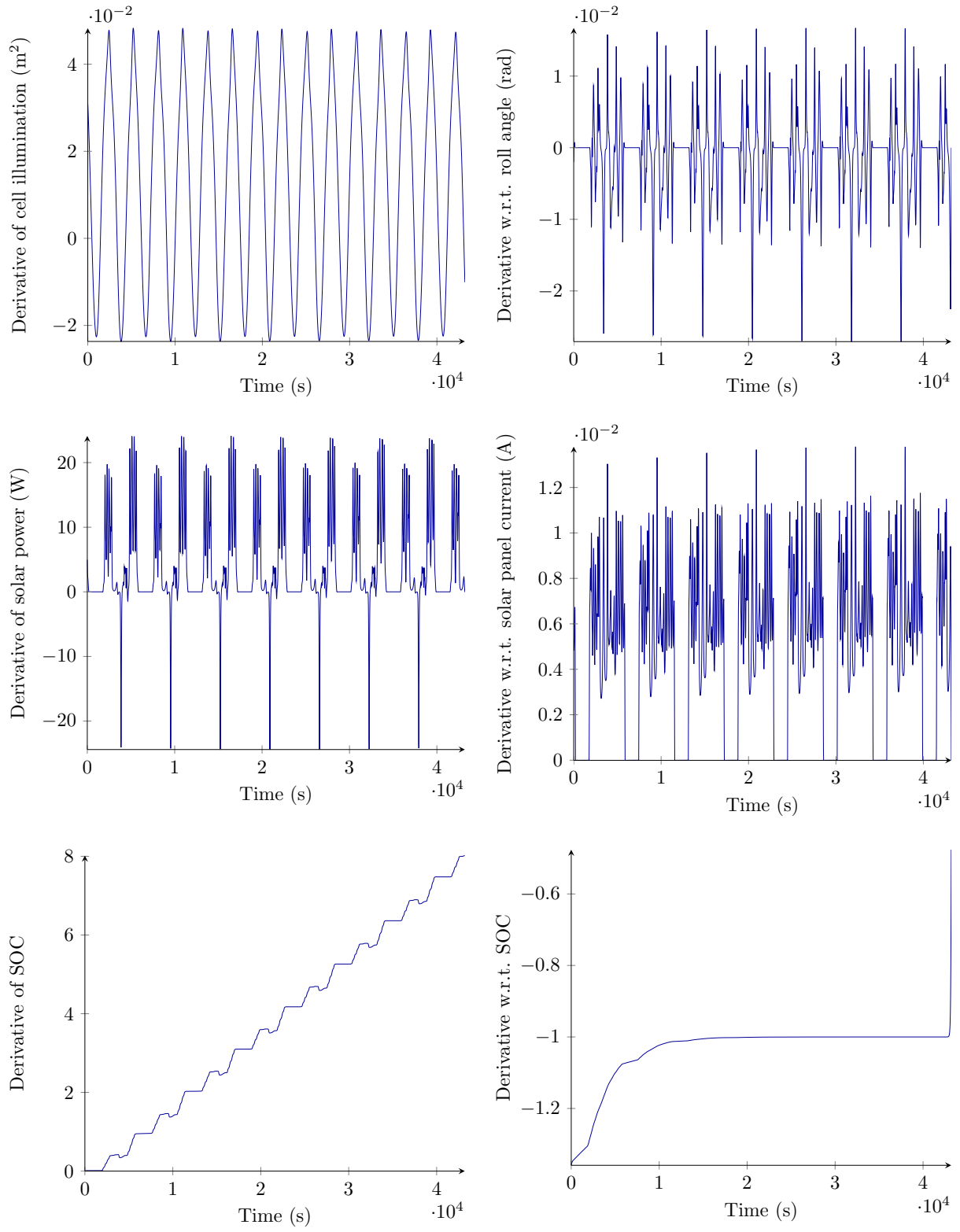

Figure 8. Derivatives of various quantities with respect to the fin angle design variable (left column) and derivatives of the minimum battery state of charge constraint with respect to various quantities (right column).

[5] Masoud Ebrahimi, Mohammad Reza Farmani, and Jafar Roshanian. Multidisciplinary design of a small satellite launch vehicle using particle swarm optimization. *Structural and Multidisciplinary Optimization*, 44(6):773–784, 2011.

[6] A.S. Fukunaga, S. Chien, D. Mutz, R.L. Sherwood, and A.D. Stechert. Automating the process of optimization in spacecraft design. In *Proceedings of the 1997 IEEE Aerospace Conference*, volume 4, pages 411–427, Aspen, CO, 1997.

[7] Roberto L. Galski, Fabiano L. De Sousa, O M. Ramos, and Issamu Muraoka. Spacecraft thermal design with the generalized extremal optimization algorithm. In *Inverse Problems, Design and Optimization Symposium*, Rio de Janeiro, Brazil, 2004.

[8] John T Hwang, Dae Young Lee, James W Cutler, and Joaquim R R A Martins. Large-Scale Multidisciplinary Optimization of a Small Satellites Design and Operation. *Journal of Spacecraft and Rockets*, 2014.

[9] A. Jafarsalehi, P. Mohammad Zadeh, and M. Mirshams. Collaborative optimization of remote sensing small satellite mission using genetic algorithms. *Iranian Journal of Science and Technology – Transactions of Mechanical Engineering*, 36(2):117–128, 2012.

[10] Saurabh Jain and Dan Simon. Genetic algorithm based charge optimization of lithium-ion batteries in small satellites. In *Proceedings of the 19th Annual AIAA/USU Conference on Small Satellites*, Logan, UT, August 2005.

[11] Hajime Kawamura, Kazuhito Naka, Norihiro Yonekura, Sanshiro Yamanaka, Hideaki Kawamura, Hideyuki Ohno, and Katsuhiko Naito. Simulation of i-v characteristics of a pv module with shaded pv cells. *Solar Energy Materials & Solar Cells*, 75:613–621, 2003.

[12] Andrew B. Lambe and Joaquim R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46:273–284, August 2012.

[13] J. R. R. A. Martins and J. T. Hwang. Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models. *AIAA Journal*, 51:2582–2599, November 2013.

[14] T. Mosher. Spacecraft design using a genetic algorithm optimization approach. In *Proceedings of the 1998 IEEE Aerospace Conference*, volume 3, pages 123–134, Aspen, CO, 1998.

[15] D. J. Richie, V. J. Lappas, and P. L. Palmer. Sizing/Optimization of a Small Satellite Energy Storage and Attitude Control System. *Journal of Spacecraft and Rockets*, 44(4):940–952, July 2007.

[16] G.M. Stump, M. Yukish, T.W. Simpson, and J.J. O'Hara. Trade space exploration of satellite datasets using a design by shopping paradigm. In *Proceedings of the 2004 IEEE Aerospace Conference*, volume 6, pages 3885–3895, 2004.

[17] Bao Zhang, Hong-Fei Teng, and Yan-Jun Shi. Layout optimization of satellite module using soft computing techniques. *Appl. Soft Comput.*, 8(1):507–521, January 2008.