SANDIA REPORT

SAND2007-5839
Unlimited Release
Printed September 2007

ASC Vertical Integration Milestone

Roscoe Bartlett (Technical PI), Scott Collis (Management PI), Todd Coffey, David Day, Mike Heroux, Rob Hoekstra, Russell Hooper, Roger Pawlowski, Eric Phipps, Denis Ridzal, Andy Salinger, Heidi Thornquist, Jim Willenbring

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2007-5839
Unlimited Release
Printed September 2007

ASC Vertical Integration Milestone

Roscoe Bartlett (Technical PI), Scott Collis (Management PI), Todd Coffey,
David Day, Mike Heroux, Rob Hoekstra, Russell Hooper,
Roger Pawlowski, Eric Phipps, Denis Ridzal, Andrew Salinger,
Heidi Thornquist, Jim Willenbring,

Abstract

The FY2007 ASC Level-2 Vertical Integration Milestone effort developed the vertical integration of many new advanced Trilinos solver packages to build new predictive solution capabilities. These were demonstrated on several relevant problems including QASPR-related semiconductor problems modeled in Charon and a MEMS design problem modeled in Aria/SIERRA. All of the objectives of the milestone have been met and exceeded in some cases. In addition to the Trilinos-specific accomplishments and the demonstration calculations, the milestone work has also helped us realize a new vision for a deeper level of collaboration between solver developers and application developers which benefits everyone involved. The bridge for this deeper collaboration is based on the foundation of nightly building and testing of the combined development versions of the application code (e.g. Charon) and Trilinos.

Contents

Figures

Tables

Summary

Overview: The ASC FY2007 ASC Vertical Integration Milestone has demonstrated a full vertical integration of numerical algorithms in Trilinos ranging from basic parallel linear algebra all the way up through transient solvers and simulation-constrained optimization in a plug an play, high-performance manner. We are targeting using optimization methods with Charon to do automatic parameter estimation against data in a fast and robust way. This work has provided a significant new capability by demonstrating the power of giving advanced algorithms better access to production application codes and by providing the software engineering infrastructure to maintain the new capabilities in the long term.

Milestone Completion: We have accomplished all of our promised objectives and have exceeded our mandate in several areas. This new capability has been released as part of Trilinos 8.0.

Selected Accomplishments: Some of the more noteworthy accomplishments achieved during the milestone work include:

1. Implemented full vertical-integration of Trilinos capabilities using standard Thyra interfaces including: parallel distributed data-structures; linear-solvers; precondtioners; nonlinear-solvers; eigen-solvers; automatic differentiation; transient solvers; and optimization solvers.
2. Demonstrated these capabilities using the ASC Charon application with QASPR semiconductor models included by performing:
   (a) Transient forward simulation,
   (b) Transient forward sensitivity analysis, and
   (c) Steady-state current-matching parameter estimation optimization.
3. As additional evidence of vertical-integration and to highlight that these algorithmic tools are ready and available for a wide range of ASC applications, we also assembled algorithms and performed:
   (a) Block eigensolves for a reacting flow problem in Charon that utilized at least eight distinct Trilinos packages.
   (b) Design optimization of a MEMS actuator using Aria/SIERRA.
4. In addition to these demonstrations, this milestone helped to highlight and address many of the challenges of injecting advanced algorithms into a production application.

1 Introduction

Many challenges exist in using cutting-edge research-driven numerical algorithms and solvers to impact challenging production-quality applications. Even more challenges exist in allowing production-quality applications to help drive the research of numerical algorithms in a significant way. Further, the challenges become more daunting when considering new predictive simulation tools like invasive uncertainty quantification (UQ), sensitivity analysis, and optimization methods. Addressing these challenges requires dedicated and sustained efforts in mathematical abstraction and design, algorithms design, application program structuring and flexibility, and ultimately good object-oriented software engineering. The FY2007 ASC Level-2 Algorithms Vertical Integration Milestone has addressed many of these challenges by demonstrating the vertical integration of many different types of numerical algorithms in Trilinos [5] and assimilating these algorithms with the application code Charon [6] in a way that can serve as a model for other algorithm-application collaborations.

On the numerical algorithms side, it is non-trivial to develop and vertically integrate state-of-the-art massively parallel numerical algorithms ranging all the way from basic linear algebra data-structures through linear and nonlinear solvers up to transient solvers and optimization. Each of these different numerical algorithm components is developed by a different team of experts to achieve highest level of quality in the implemented algorithms in a way that a single team of non-expert developers can not match. This effort is further complicated by the stringent parallel scalability requirements that are part of capability computing [8].

To address the vertical integration problem, a major thrust of this milestone was to further develop and demonstrate the Thyra interface layer in Trilinos as a way to seamlessly integrate and plug-and-play different implementations of various numerical solver components, each developed by different teams of expert algorithm developers. In short, the goal of Thyra within Trilinos is to cleanly support nearly any vertical integration of numerical solvers that makes sense mathematically in a scalable and maintainable way.

To demonstrate vertical interoperability and integration, we focused primarily on the Charon [6] application code with an emphasis on semiconductor problems related to the QASPR [7] program. In particular, we targeted sensitivity analysis and parameter estimation optimization problems. As an indication of the power and generality of these algorithms, we also demonstrated some of the developed capabilities on reacting flow problems in Charon and on coupled thermal/electrical/structural problem in Aria/SIERRA.

The bridge between numerical algorithms and application codes is software. Numerical algorithms must be implemented in software and great effort and care is required to achieve maximum robustness, speed, and scalability. Integrating numerical software into application codes is often difficult and tedious work and keeping the numerical solver software and the application software up to date with each other is critical to allow for the flow of ideas and capabilities back and forth between algorithm developers and application developers. Good software engineering practices are needed to keep the software “bridge” between algorithms and applications alive and productive for all involved.

During the course of this milestone work, we have shown how to scalably manage the vertical integration of advanced numerical algorithms from basic linear algebra, linear solvers, nonlinear

solvers, stability and bifurcation methods, automatic differentiation, transient solvers, and simulation-constrained optimization. We have demonstrated various instances of vertically integrated solvers with several different problems modeled in production applications. We have also developed a software integration process and infrastructure between Charon and Trilinos through which algorithm developers and application developers can more closely work together and yield faster, more robust, and better tailored numerical algorithms. This is done in a way that better meets the specific needs of important application customers while at the same time stimulating publishable numerical algorithm research.

2 Trilinos algorithms and vertical integration with Thyra

A primary goal of this work was to drive the development and vertical integration of the algorithms in the packages Belos (block linear solvers), Anasazi (block eigensolvers), Rythmos (time integrators), and MOOCHO (optimization) in order to implement advanced solver/analysis/optimization capabilities. In addition, we incorporated several other Trilinos packages including Epetra (linear algebra data structure packages), Ifpack (incomplete factorization preconditioners), ML (multi-level algebraic preconditioners), Amesos (direct sparse solvers), NOX (nonlinear equation solvers), LOCA (stability and bifurcation methods), Sacado (automatic differentiation) and more.

Vertical integration of numerical algorithms refers to the nesting of numerical solvers within each other to enable the construction of sophisticated multi-component solver capabilities. For example, a modularized ODE/DAE transient sensitivity solver such as is implemented in Rythmos defining a transient optimization problem to be solved by MOOCHO, shown in Figure 1, demonstrates at least eight different levels of vertical algorithm integration. It is critically important that these types of modular solvers be constructed in a way that allows for almost any individual algorithm/solver component to be swapped out for a version better suited to a given problem. For example, while an implicit backward Euler time stepper may be appropriate for one particular class of problems, a higher-order implicit time stepper, such as a variable-order variable step size BDF method, may be superior for another important class of problems. The same need for flexibility and modularity is required for the selection of preconditioners, linear solvers, and other numerical algorithm components. Without this type of flexibility, application developers may be justified in writing their own, often suboptimal, algorithms due to an inability to change an inappropriate component of a more general solver (there are many examples of this in production codes). For example, there was a production circuit simulation application code that was not able to reuse an existing well known time integration solver due to the inability to redefine the nonlinear solver used for the time step equation.

Integrating independently developed numerical algorithms and software in an efficient and manageable way requires the development of standard object-oriented interfaces. Typical approaches for combining N different numerical solver algorithms together may require up to N^2 different 1-to-1 specific connections in the worst case. Standard interfaces break the non-scalable N-to-N dependencies between different numerical software components into separate scalable 1-to-N dependencies.

Standard interfaces for these types of numerical algorithms have been developed and refined in the Trilinos package *Thyra*[1]. Thyra is comprised of a layered set of abstract C++ interfaces to linear operators and vectors, preconditioners and linear solvers, an interface to nonlinear models called the ModelEvaluator, and a nonlinear equation solver interface. Layered on top of the Thyra interfaces are Rythmos interfaces defining a number of higher-level abstractions for time stepping and time integration algorithms. All of these interfaces are built on the concept of Abstract Numerical Algorithms (ANA) where only the essential mathematical properties of the objects are considered without any implementation specific details [1]. This high-level ANA approach allows a for a level of generality, reuse, and efficiency that is not possible with other types of approaches.

Of particular significance to this milestone is the use of the Thyra nonlinear ModelEvaluator

[1]The word *Thyra* means “interface”, or “grand entrance” in Greek.

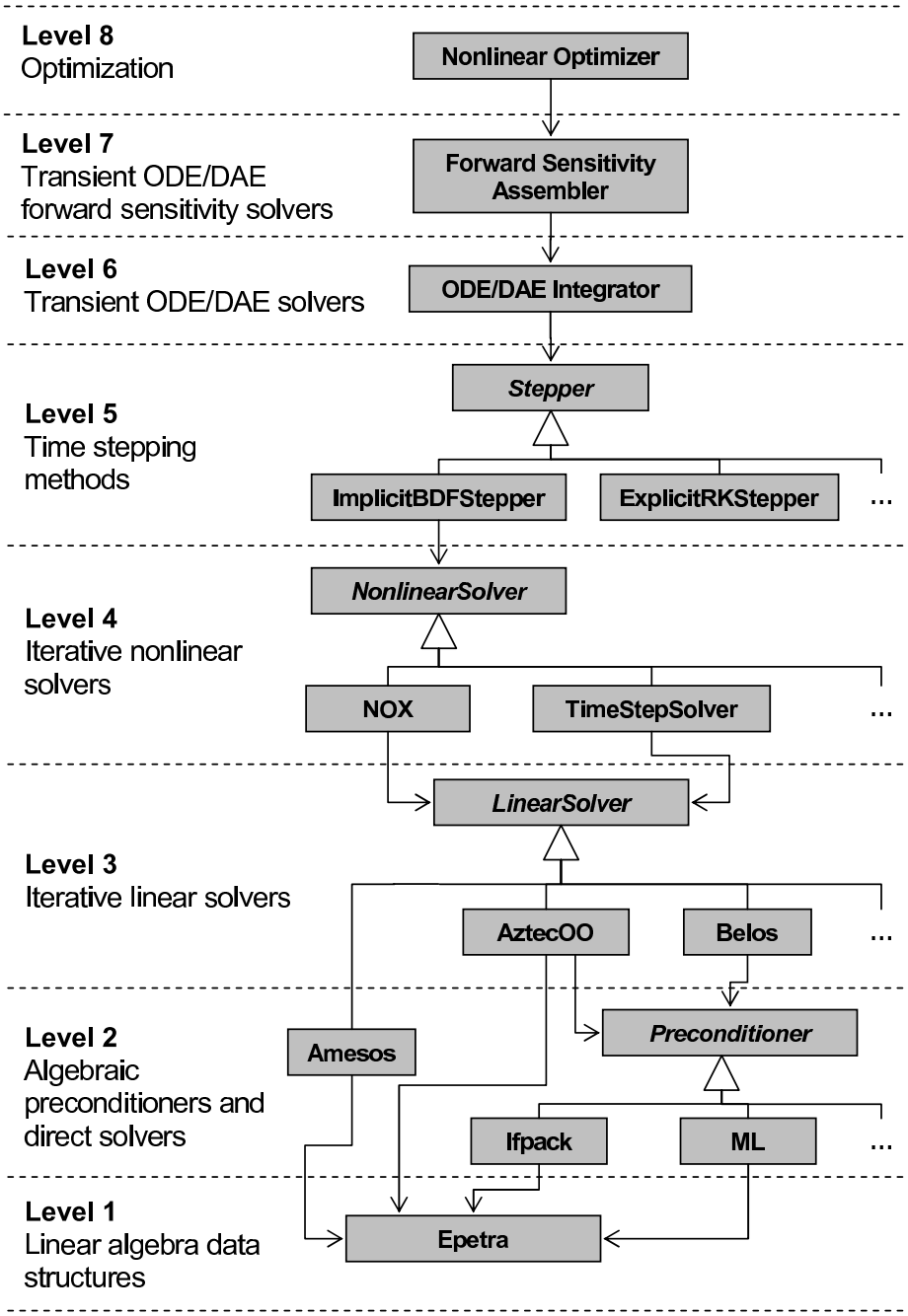

Figure 1. Example of eight different levels of vertical integration that exist in a transient ODE/DAE sensitivity solver in Rythmos with MOOCHO optimization.

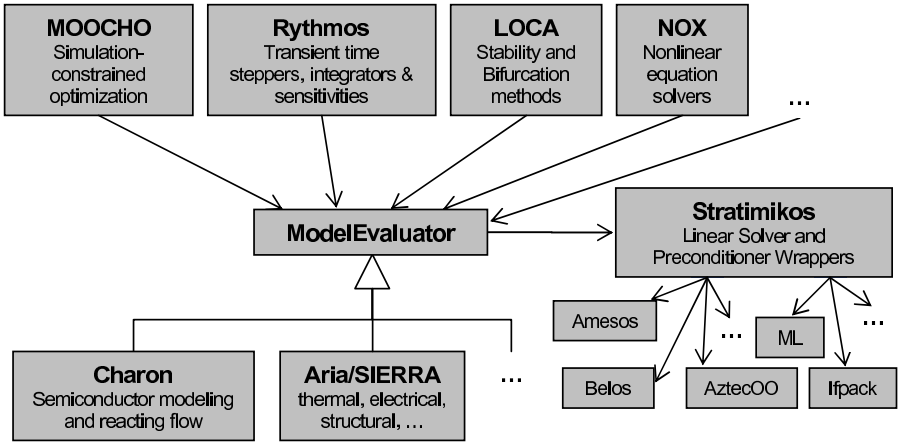

Figure 2. Shows how the Thyra ModelEvaluator and Stratimikos linear solver interfaces relate to nonlinear abstract numerical algorithms (ANAs) and nonlinear applications (APPs). This demonstrates the decoupling and scalability of the Thyra standard interface approach.

interface and supporting software. Through a single interface, a variety of nonlinear problems are presented to NOX, LOCA, Rythmos and MOOCHO using both Charon and Aria/SIERRA as shown in Figure 2. The *Stratimikos* component shown in Figure 2 is a Trilinos package built on top of the Thyra preconditioner and linear solver interfaces that provides unified parameter-driven access to a great number of preconditioner and linear solver capabilities in Trilinos. Let N and M be the number of nonlinear algorithms and applications, respectively, that one wishes to use with each other. Without a standard interface like the Thyra ModelEvaluator, it would take $N \times M$ different 1-to-1 interfaces to link all M applications and N nonlinear algorithms. The ModelEvaluator and Stratimikos approach is a clear demonstration of breaking up these N-to-M dependencies which results in scalable 1-to-N and 1-to-M dependencies. Adding a new nonlinear ANA solver only requires a single set of Thyra interfaces and then this solver can be used by the entire set of application codes that support the needed functionality. Likewise, an application just needs to implement the single ModelEvaluator interface, and then it can access all of the various supported nonlinear solvers. Finally, a new preconditioner or linear solver can be wrapped under Stratimikos and then be available to all applications and nonlinear ANAs that support the ModelEvaluator interface. This type of reuse and interoperability for massively parallel nonlinear applications and algorithms to this degree has perhaps never before been achieved and this type of reuse and interoperability will only increase in the future.

This milestone work clearly highlights the effectiveness of the Thyra interface layer and ANA approach as demonstrated by the variety of the different types of vertically integrated solver configurations that were achieved and the types of numerical problems that were solved. Specific examples of different vertical solver integrations and problems solved are given in Section 5.

3 Charon/QASPR

The Qualification Alternatives to the Sandia Pulsed Reactor (QASPR) program is focused on future qualification of electrical weapons systems in short pulse neutron environments (SPNE). With the shutdown of the SPR III reactor in 2006, full system failure tests in SPNE are no longer possible. Replacement of this qualification testing is critical to the future of nuclear weapons certification. The QASPR program is targeting this gap with a combination of testing at alternative test facilities, such as the Annular Core Research Reactor (ACRR) and the Ion Beam Laboratory (IBL), with high fidelity predictive computational modeling. As well as supporting qualification, these tools will increase the designer's capability to develop robust systems during the early stages of systems development.

The ASC Charon semiconductor device simulator is a finite element tool capable of simulating high fidelity models of stockpile devices using large scale ASC platforms for unprecedented scales and fidelities of transient displacement damage, due to neutrons, in stockpile devices such as power bipolar junction transistors. This capability was developed using predictive fundamental physics models developed at SNL through experimental and atomistic modeling (Density Functional Theory and Molecular Dynamics) and massively parallel algorithmic technology in the ASC Trilinos solver libraries and the ASC Nevada finite element framework.

A critical aspect of the QASPR problems of interest is the refinement of physics models for un-irradiated and irradiated devices in comparison to experimental data. Two key examples in the current QASPR process include: 1) calibration of the device doping profiles based on secondary ion mass spectrometry, which has a substantial absolute error for density (approximately a factor of two), and electrical characterization of the device; 2) calibration of displacement defect physics parameters, which have an order of magnitude uncertainty, against transient pulse electrical characterization experiments. Currently, these "calibrations" are accomplished with substantial manual effort and resources involving hundreds to thousands of calculations. The new algorithmic technologies integrated in Charon show great promise in their ability to efficiently and robustly solve these problems with dramatically less manpower and computational resources.

4 Application and Trilinos development nightly integration

Through the course of the milestone work, we found that in order to keep moving forward and avoid backslides in capability (which happened early on), we implemented nightly building and testing of the development versions of Charon and Trilinos. Every night, we take what is in the Charon and Trilinos development repositories and build the combined Charon & Trilinos application and run a large set of regression tests. In the time since we started nightly building and testing, the number of tests in the Charon test suite has gone up from under 50 to over 100, and *30 of the new tests are directly related to this milestone work.*

In the execution of this nightly building and testing process, we have learned many things about how to do continuous integration [4] of application and algorithms (Trilinos) software and we have realized many important unplanned benefits. This will allow us to reap many more benefits in the future if this process is maintained and extended. There are both production-related benefits and research-related benefits that help both the application developers and the algorithm developers to achieve their goals. On the research side, this significantly reduces the overhead required for algorithm developers to try their algorithms out on production quality problems. Developing a numerical solver with a production problem exposes the algorithm developer to a whole host of issues (e.g. poor scaling, ill-conditioning, difficult convergence, etc.) that are hard to replicate in model problems. On the production side, constant integration insures that the application and Trilinos are always up to date and satisfying the application's requirements. Therefore, when it is time for a release, only a final set of acceptance tests are required and then the codes can be branched and released shortly after. This helps to reduce a whole host of risks such as slipped schedules and broken features[2].

We have seen several different scenarios over the years where this nightly building and testing infrastructure would have facilitated collaboration between application and algorithm developers for the advantage of both.

For example, suppose an application developer is running a new problem and discovers some strange behavior from a numerical solver. The algorithm developer may look at the results and speculate what the cause of the behavior might be or if a different variation of the algorithm might help. However, if the algorithm developer is stuck having to use a released version of Trilinos, it will be more difficult to make any major changes in order to investigate the behavior. Also, there may already be improvements made to the algorithm in the development branch of Trilinos that may be able to address the problem. Without the infrastructure of nightly building and testing, it may not be cost effective or practical to bring the development versions of the application and Trilinos up to date in order to try the updated algorithm. The algorithm and application developers may have to wait until the next major release of Trilinos before the new algorithms can be tried. This time delay works to no one's advantage and can kill the collaboration. With nightly building and testing in place, an easy path for collaboration is maintained where the Trilinos algorithm developer can try their latest and greatest algorithms on these types of challenging production problems and the information learned from trying to solve these production problems can feed back into algorithm development. To some extent, this back and forth already happens but without a foundational process in place to streamline it, this bidirectional flow is greatly restricted.

Another example where nightly building and testing of the development versions of the application

[2] Also known as regressions

and Trilinos would ease the path for collaboration is when an application developers wants to try a new capability in Trilinos without a lot of work[3]. When the up-to-date development version of Trilinos is available from within an application, a new algorithm or capability from Trilinos can be accessed much more easily. Typically, even a small increase in the overhead needed to try out a new Trilinos capability in an application can be enough to kill a potentially fruitful collaboration between an application and algorithm developer. Nightly building and testing of the development versions of the application and Trilinos removes a unexciting (and therefore demoralizing) but critical obsticle to collaboration and impact.

Nightly building and testing of an application code and Trilinos brings algorithm developers and application developers closer together – exchanging ideas and concerns – and refocuses Trilinos developers on customer efforts while still helping drive publishable numerical algorithm solver research and reduces barriers for new algorithms to have impact through production application codes.

[3]This example really happened and it was the inability to readily access the development version of Trilinos within the application that killed the collaboration

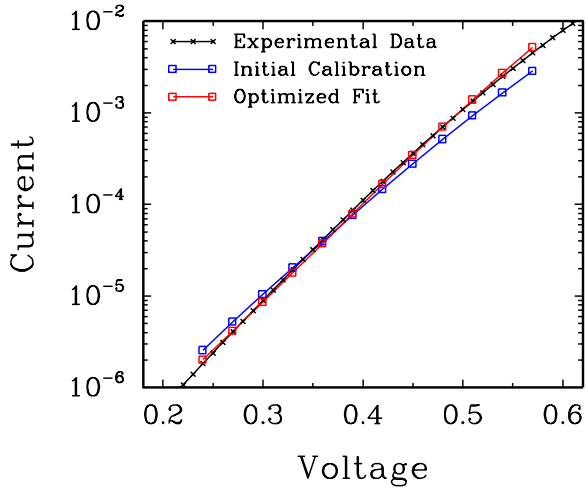

Figure 3. This figure illustrates how multi-point optimization can be used to optimize system parameters so that a *curve* of responses can best be matched. In this case, 12 steady-state current predictions, running in parallel, were fit to the experimental data.

5 Demonstration solver vertical integrations and calculations

While many different numerical solver configurations were used to solve a variety of different numerical problems in Charon and Aria during the milestone work, some of the more noteworthy examples are given below. For each example, the list of vertically integrated algorithm packages that were linked together to solve the problem are given. Note that the Thyra package was used in all cases as the standard interface to pull these algorithm packages together.

5.1 Steady-state semiconductor current-matching parameter estimation problems

Vertically integrated packages: MOOCHO, Stratimikos, Belos, Ifpack, Thyra, and Epetra

We solved both single-point and multi-point inverse optimization problems for the QASPR semiconductor model 5614. Here, we used MOOCHO to solve a simulation-constrained least-squares optimization problem to minimize the deviation between the simulated current through the device and the target current by manipulating the poorly known defect reaction parameters. We showed significant improvement in accuracy and speedup over non-invasive block-box methods.

The multi-point algorithm allows parameters to be matched over the whole current-voltage curve simultaneously with excellent scalability by using another dimension of parallelism over the multiple data points. This required development in EpetraExt and modifications to the Nevada/Charon code where multiple instances of the code run simultaneously with partitioned MPI Communicators, while the linear algebra and optimization algorithms operate on a global

MPI communicator.

As seen in Figure 3, we were successful in having the multi-point algorithm find the optimal parameter value so that a curve of 12 points would best match the experimental data. Since the application for this problem specification runs well in serial, this was run on 12 processors of Thunderbrid for near-optimal scalability. The results shown were computed by altering the total amount of radiation damage, and then optimally fit using the voltage at the second contact as the optimization parameter.

Unfortunately, we were not able to match all experimental data to high precision by just using the most physically relevant parameters that govern the defect physics as identified by the QASPR modelers. While we were not able to provide a definitive proof of the cause, single-point current-matching numerical experiments suggest that many of these experimental states are simply not reachable given the other model operating parameters which are supposed to be known to high precision. This suggests that either there was experimental error, or the other operating parameters did not really match the physical system, or the model was incomplete in some way. The main point is that while we were not able to diagnose the cause of the problem, the milestone has developed some of the tools needed to help answer these questions.

One interesting output of this work was the behavior of the optimization algorithms in MOOCHO and the basic forward Newton solver algorithms. It turned out that the radiation defect physics steady-state semiconductor model caused many convergence challenges for the algorithms. In some cases, even small perturbations in the defect reaction parameters would cause the forward Newton's method to fail to re-converge the state equations and the optimization algorithms in MOOCHO suffered similar convergence difficulties. A problem of this type provides an exciting opportunity for further research into globalization methods for simulation-constrained optimization. Typically, numerical algorithm researches are resigned to develop their algorithms on "model" problems that they try to make difficult but it is hard to reproduce the kinds of unexpected challenges that are manifested in real production-quality applications. Now that these test problems are part of the Charon test suite, they can be preserved and will provide easy access for further algorithm research. Without nightly testing, it is our past experience that these types of interesting test problems always fall away due to code and other changes that break the connection between the production application and the numerical algorithm software.

5.2 Transient QASPR semiconductor forward simulation

Vertically integrated packages: Rythmos, NOX, Stratimikos, Belos, Ifpack, Thyra, and Epetra

We used a high-order accuracy-controlling implicit BDF integrator algorithm in Rythmos (along with other mentioned vertically integrated packages) to solve forward simulation problems for the 2-dimensional prompt neutron physics QASPR 2n2222 semiconductor model. This capability demonstrates the high accuracy time integration capability for a critical QASPR problem. Robust and efficient solution of this transient simulation allows us to quantify the effects of prompt neutron irradiation for a key stockpile device in direct support of the QASPR mission.

5.3 Transient QASPR semiconductor sensitivities

Vertically integrated packages: Rythmos, NOX, Stratimikos, Belos, Ifpack, Thyra, and Epetra

We used the new forward sensitivity solver in Rythmos to demonstrate the calculation of transient sensitivities of the electric current in a Charon simulation of the bipolar junction transistor subject to radiation damage with respect to 40 model parameters. The parameters of interest in this calculation are physics parameters associated with damage mechanisms in the device which dramatically impact the transient electrical performance of the bipolar junction transistor. As one of the devices used in the stockpile, it is critical to characterize its performance. Without underground testing and now the decommissioning of fast pulse neutron facilities such as the Sandia Pulsed Reactor (SPR), a combination of alternate experimental data such as ion beam data and predictive modeling is critical to the future of weapons systems qualification. The nominal values of these parameters have an order of magnitude uncertainty making best fit plus uncertainty untenable for the customer. Through optimization and calibration of the Charon model to experimental data, the uncertainty in these parameters can be substantially reduced. This transient sensitivity analysis not only facilitates gradient based optimization technology but also allows detailed analysis of the mechanisms and their importance in relation to the critical device performance metrics guiding future improvements to the model.

The 40 parameters used in the sensitivity analysis are the reaction cross-sections and activation energies of the radiation defect reactions shown in Table 1. Sensitivities of the base current at several times after the radiation pulse are shown in Figure 4, clearly demonstrating which physics the current is most sensitive to. Transients plots of the base current sensitivity with respect to two of these important parameters are shown in Figure 5.

The typical approach for obtaining these sensitivities is through non-invasive finite-difference methods. We compared computing sensitivities using first-order finite-differencing to the direct method in Rythmos and found several dramatic advantages of the Rythmos approach. First, because perturbing parameters ultimately will cause different time steps to be taken, it is only possible to compute finite-difference sensitivities at selected times by setting breakpoints in the time integrator. However the Rythmos approach provides full sensitivity values at all time steps as shown in Figure 5. Moreover the variation in step size with respect to parameter perturbation adds significant noise to the sensitivities computed by finite-differencing, making them quite inaccurate and not suitable for gradient-based optimization. Figure 6 displays a comparison of the direct sensitivities to first-order finite-differences and shows that tighter time integration tolerances are required to reduce this noise to get even order-of-magnitude accuracy from the finite-difference calculation. **Finally the transient sensitivities for all 40 parameters using Rythmos were obtained in significantly less computing time, less than 1/5th of the time required by first-order finite-differencing**.

5.4 Block eigen-solves for reacting flow problem

Vertically integrated packages: LOCA, NOX, Anasazi, Stratimikos, Belos, ML, Ifpack, Amesos, Thyra, and Epetra

We successfully demonstrated large-scale eigenvalue calculations using block eigensolvers on the

Table 1. Defect reaction parameters for transient current sensitivity analysis of the 2n2222 bipolar junction transistor problem

Reaction Cross-Section Parameters					
Index	Value	Reaction	Index	Value	Reaction
1	3.0e-16	$e^- + V^- \rightarrow V^{--}$	15	1.8e-13	$h^+ + V^- \rightarrow V^0$
2	3.0e-16	$V^{--} \rightarrow e^- + V^-$	16	1.8e-13	$V^0 \rightarrow h^+ + V^-$
3	1.8e-14	$e^- + V^0 \rightarrow V^-$	17	3.0e-15	$h^+ + V^0 \rightarrow V^+$
4	1.8e-14	$V^- \rightarrow e^- + V^0$	18	3.0e-15	$V^+ \rightarrow h^+ + V^0$
5	3.0e-14	$e^- + V^+ \rightarrow V^0$	19	3.0e-16	$h^+ + V^+ \rightarrow V^{++}$
6	3.0e-14	$V^0 \rightarrow e^- + V^+$	20	3.0e-16	$V^{++} \rightarrow h^+ + V^+$
7	3.0e-14	$e^- + V^{++} \rightarrow V^+$	21	3.0e-16	$h^+ + B^- \rightarrow B^0$
8	3.0e-14	$V^+ \rightarrow e^- + V^{++}$	22	7.5e-14	$B^0 \rightarrow h^+ + B^-$
9	3.0e-16	$e^- + P^+ \rightarrow P^0$	23	3.0e-14	$h^+ + PV^- \rightarrow PV^0$
10	1.5e-13	$P^0 \rightarrow e^- + P^+$	24	3.0e-14	$PV^0 \rightarrow h^+ + PV^-$
11	3.0e-15	$e^- + PV^0 \rightarrow PV^-$	25	1.0	$V^{--} + P^+ \rightarrow PV^-$
12	3.0e-15	$PV^- \rightarrow e^- + PV^0$	26	1.0	$V^- + P^0 \rightarrow PV^-$
13	3.0e-14	$h^+ + V^{--} \rightarrow V^-$	27	1.0	$V^- + P^+ \rightarrow PV^0$
14	3.0e-14	$V^- \rightarrow h^+ + V^{--}$	28	1.0	$V^0 + P^0 \rightarrow PV^0$

Reaction Activation Energy Parameters					
Index	Value	Reaction	Index	Value	Reaction
29	0.09	$V^{--} \rightarrow e^- + V^-$	35	1.03	$V^- \rightarrow h^+ + V^{--}$
30	0.40	$V^- \rightarrow e^- + V^0$	36	0.72	$V^0 \rightarrow h^+ + V^-$
31	1.07	$V^0 \rightarrow e^- + V^+$	37	0.05	$V^+ \rightarrow h^+ + V^0$
32	0.99	$V^+ \rightarrow e^- + V^{++}$	38	0.13	$V^{++} \rightarrow h^+ + V^+$
33	0.045	$P^0 \rightarrow e^- + P^+$	39	0.045	$B^0 \rightarrow h^+ + B^-$
34	0.44	$PV^- \rightarrow e^- + PV^0$	40	0.68	$PV^0 \rightarrow h^+ + PV^-$

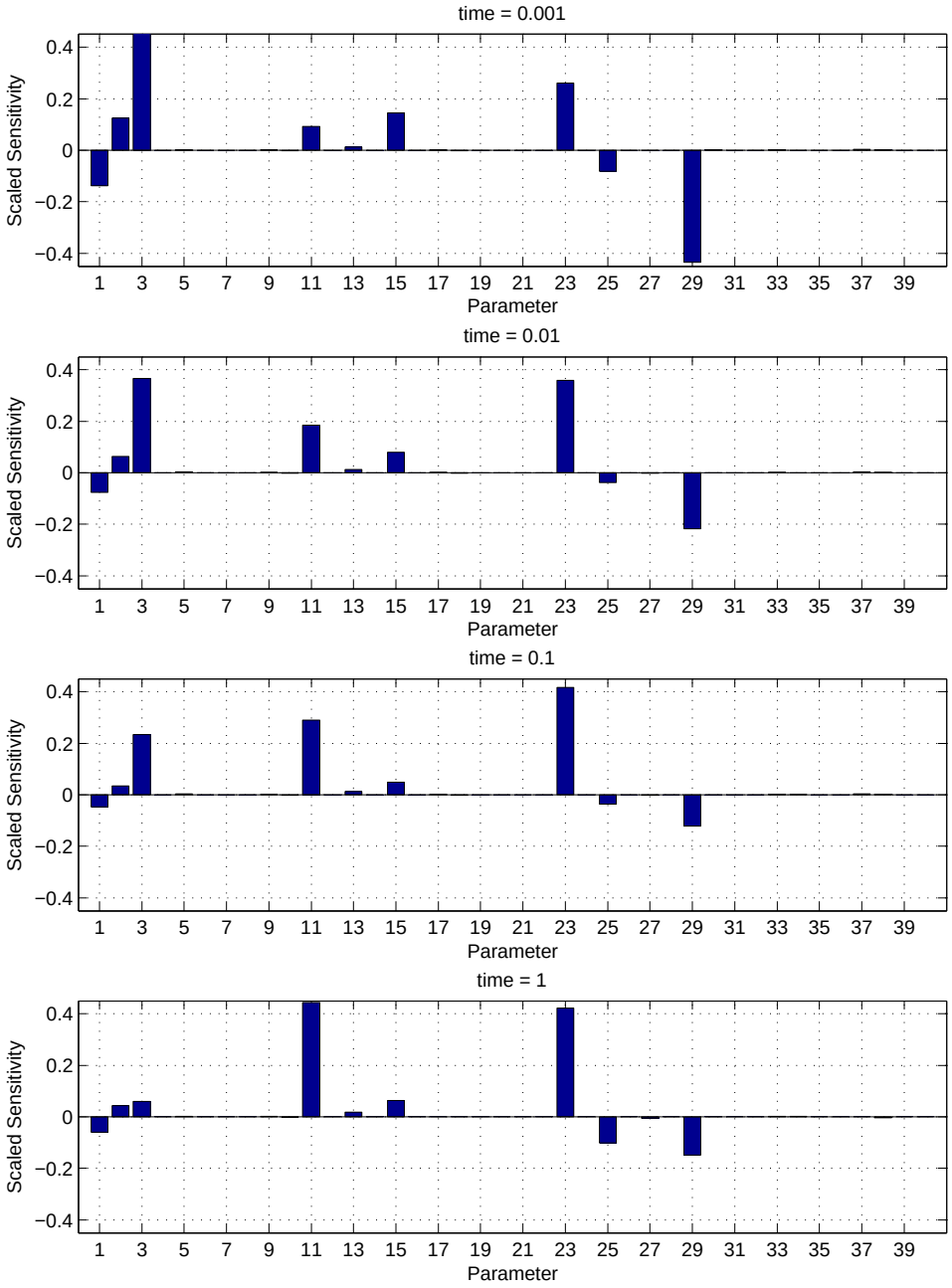

Figure 4. Scaled transient base current sensitivities at selected times of the 2n2222 device. The parameters corresponding to each number are displayed in Table 1. Sensitivities are scaled to $(p/I)(dI/dp)$ where p is the parameter value, I is the current, and dI/dp is the unscaled sensitivity.

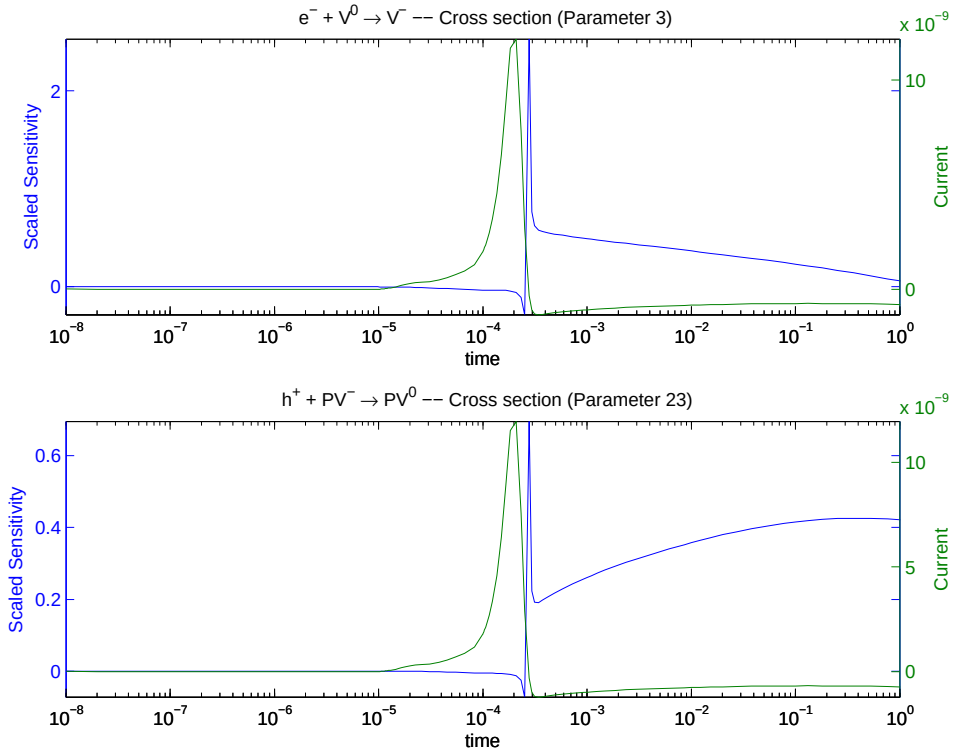

Figure 5. Transient history of sensitivities 3 and 23 from Figure 4 (blue curves) and base current (green curves).

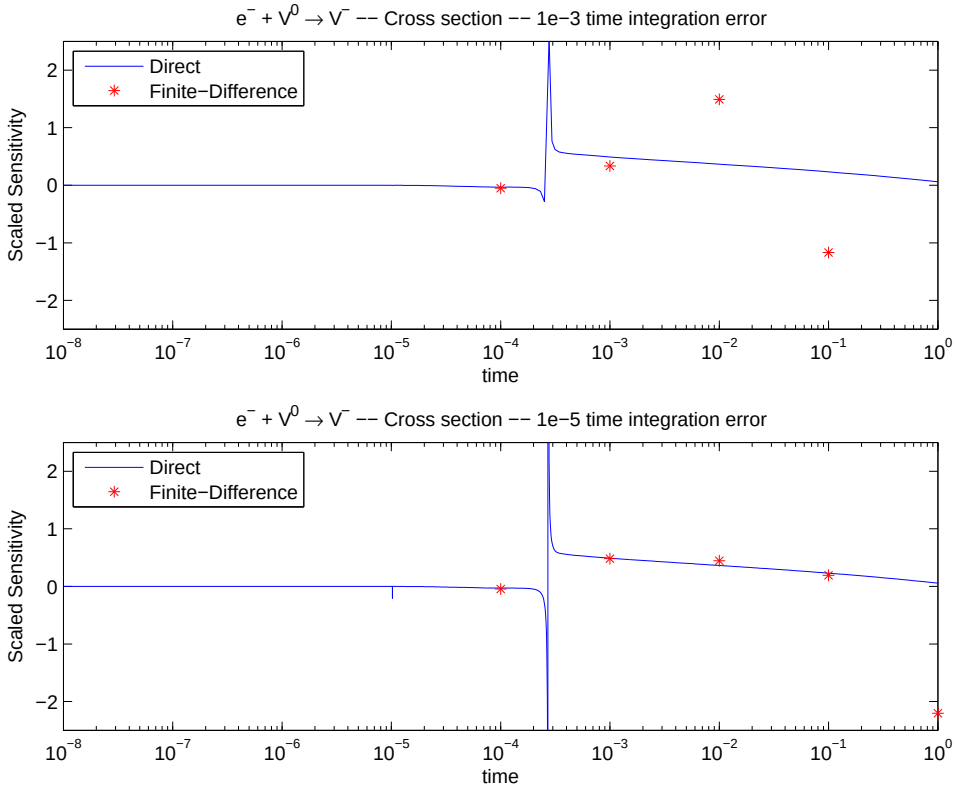

Figure 6. Comparison of direct Rythmos sensitivities (blue curves) with first-order finite-differences (red stars) for parameter 3 with two time integration error tolerances, 1e-3 and 1e-5. Tighter time integration tolerances are required to get even order-of-magnitude correctness out of finite-difference approaches.

Table 2. Leading eigenvalues as a function of Rayleigh number for Hydromagnetic Rayleigh-Bernard problem.

	Rayleigh Number			
Index	2020	2030	2040	2050
1	-0.13413	-0.062685	0.0086453	0.079866
2	-0.15978	-0.086379	-0.013056	0.060184

hydromagnetic Rayleight-Bernard (HRB) problem. The HRB problem is important for verification of magnetohydrodynamics physics due to the existence of analytic solutions published by Chandresekhar [2]. The problem consists of a fluid sandwiched between upper and lower walls. The lower wall is heated and the upper wall is cooled. As the temperature gradient between the walls is increased (an increase in the Rayleigh number), a buoyancy driven instability occurs that transitions the fluid from a quiescent (no-flow) state to a recirculating (non-zero flow) state with repeating cells of rotating fluid. Figure 7 shows the flow solution for the recirculating state.

The onset of this instability (a pitchfork bifurcation point) can be located by monitoring the leading eigenvalues during a parameter continuation. The instability occurs when the real part of the leading eigenvalue crosses the imaginary axis (e.g. becomes zero). Therfore to locate an exchange of stability, we watch for the leading eigenvalue to switch from a negative value to a positive value as the parameter continuation is performed. The Rayleigh number (Ra) is the chosen continuation parameter and the magnetic field strength is fixed to a constant value. After the bifurcation, the no-flow solution becomes unstable. Table 2 show the leading two eigenvalues as a function of the Reyleigh number. We observe that the eigenvalues flip sign between 2030 and 2040. Figure 8 shows the computed eigenvectors for the velocity at a Rayleigh number of 2040, just past the bifurcation point at 2039. This figure depicts the perturbation to the base no-flow solution that forces the transition to the recirculating state.

The problem was discretized into 5 million elements leading to a linear system with 5 million unknowns that was solved using 256 processors on Thunderbird. We used LOCA/NOX to solve the system of fully coupled nonlinear equations for a variety of Rayleigh numbers using arc-length continuation. At the end of each continuation step, LOCA calls Anasazi to compute the eigenvalues and eigenvectors. The eigenvalue computation used Belos to perform Block and Pseudo-Block GMRES on the linear systems. ML supplied the preconditioning for the linear solves. ML internally used a 3 level V-cycle, calling IFPACK with ILU for the smoothers and Amesos (KLU) for the direct solve on the coarse grid.

The eigenvalue calculation demonstrates a general capability through coupled Trilinos packages to perform large-scale block eigenvalue calculations. This will be a critical ASC capability for performing stability and bifurcation analysis.

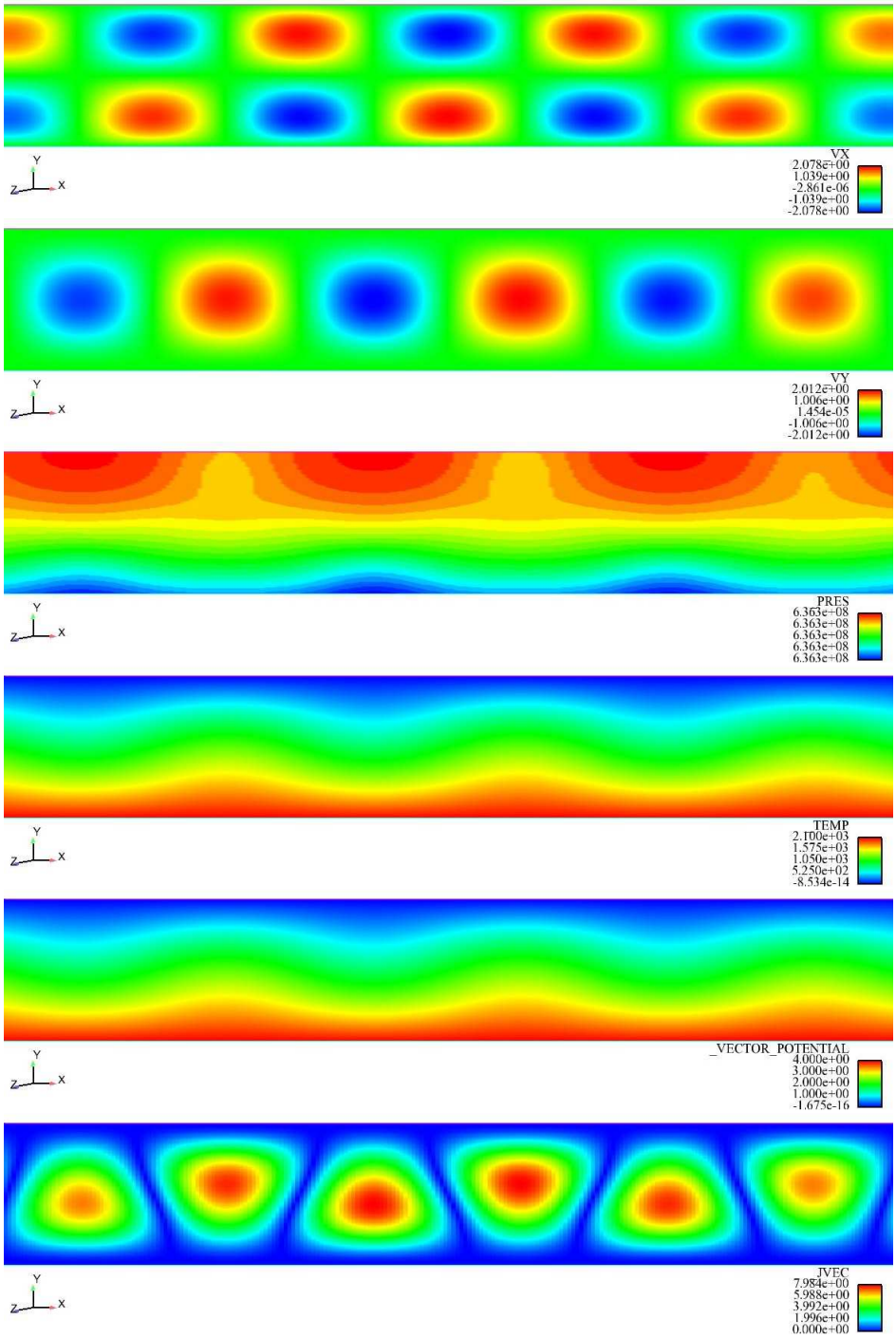

Figure 7. Plot of the solution for a Rayleigh number of 2100 and magnetic field strength of 16 on the non-zero flow branch of the pitchfork bifurcation.

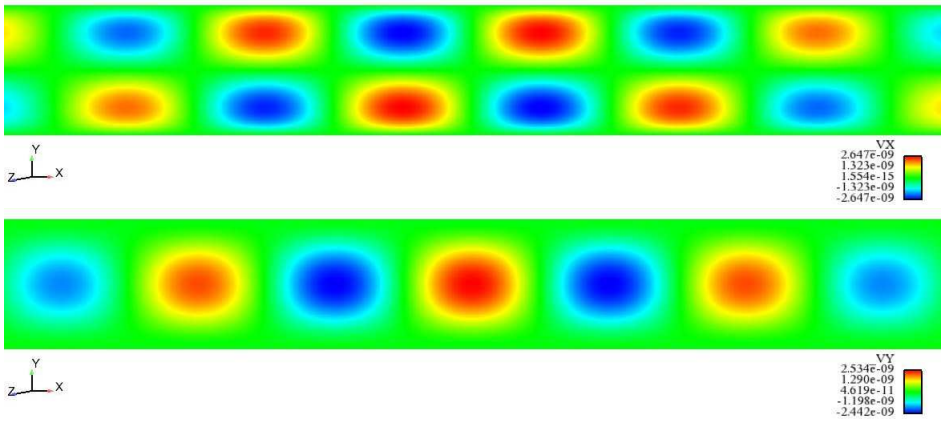

Figure 8. Plot of the eigenvectors for the x and y velocities. This perturbation corresponds exactly to the flow solution above.

5.5 Design optimization problem with Aria/SIERRA

Vertically integrated packages: MOOCHO, Stratimikos, Belos, Ifpack, Thyra, and Epetra

We have demonstrated the minimally invasive optimization algorithm in MOOCHO on a MEMS actuator problem in Aria/SIERRA. In this problem, an applied voltage across the Silicon Carbide beam causes resistive heating, which in turn causes thermal expansion, which in turn causes the beam to deflect upwards. The optimization problem was formulated as follows: find the value of the applied voltage parameter so that the beam deflection most closely matches a given design value (e.g. a deflection distance of 0.05). The proof-of-concept was successful, with the optimization problem solving to 8 digits of accuracy (Figure 9). This demonstration also highlights the speed of the invasive algorithms, as it solved the optimization problem in only twice the time of solving a single steady-state calculation. We experimented briefly with black-box finite-difference optimization methods which are commonly used to solve simulation-constrained optimization problems. These methods are known to be very sensitive to step perturbation sizes and numerical parameters and we were not able to make any progress in the solution to even compare performance of these different optimization approaches.

This Aria application demonstrates that the algorithms and software are general and immediately applicable to ASC applications other than Charon. The development of the ModelEvaluator interface will pave the way for more Trilinos capabilities, such as Rythmos, to be assimilated into Aria. Since the model evaluator was implemented at the Sierra Solution Control level, there is a now a direct path for all Sierra applications (particularly implicit codes such as Adagio) access to all Trilinos analysis algorithms.

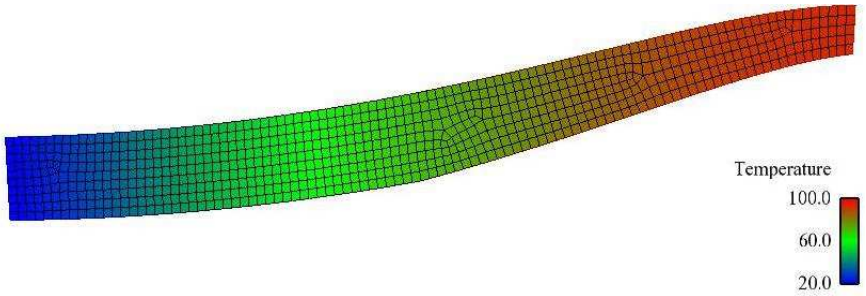

Figure 9. Plot of the final Aria solution showing the deformed beam and Temperature contours. The optimization algorithm found that an applied voltage of 1.934 would cause the beam to exactly match the design criterion of having the top corner of the beam deflect a distance of 0.05 at steady-state.

Table 3. Typical magnitudes of variables and functions in the semiconductor optimization problem which span a range of 10^{25}!

State Variables	10^{0} to 10^{8}
State Constraints	10^{-4} to 10^{10}
Parameters	10^{-15} to 10^{0}
Currents	10^{-8}

6 Numerical Issues

In order to be able to solve several of the demonstration calculations (especially the optimization problems), we had to address various numerical issues including matrix singularities and scaling. Without addressing all of these issues, many of the demonstration problems were essentially unsolvable with 64 bit floating point numbers.

The first issue that needed to be addressed was that the semiconductor reaction defect physics model gave a nearly structurally singular Jacobian for the steady-state model. A singular Jacobian makes the optimization problems unsolvable. In order to address the singularity problem, an automated way of reformulating the equations involving immobile species was devised. To identify the redundant equations, an SVD is performed on the reaction mechanism incidence matrix. From this SVD, a new reduced reaction mechanism is produced which is used to build the Charon input file.

The other major numerical issue that had to be addressed was the scaling of all of the quantities. The typical magnitudes of various quantities seen in the steady-state single-point current-matching optimization problem are shown in Table 3. The magnitudes represented in Table 3 span a range of 10^{25}! This range of scalings will break almost any numerical method implemented with 64 bit float point numbers.

In order to solve the steady-state current-matching parameter-estimation optimization problems, we had to scale every quantity including state variables, state constraints, optimization parameters, and currents. Various strategies were used to find reasonable scalings for each of the quantities and in the end we ended up with all variables and functions scaled to a size of about 1.0. Even with these scalings in place, some of the intermediate derivatives were of size 10^{6} and the condition number of the state Jacobian was 10^{5}. With all of these quantities scaled, the optimization algorithms performed reasonably well and were able to invert for the parameters to a high precision (8 digits of accuracy in current and parameter matches in many cases). If even one of to scalings was removed, the optimization algorithms failed to find the solutions.

Optimization algorithms, more than any other type of algorithm, are more sensitive to scaling

issues because they must make comparisons of quantities that are completely unrelated. All-at-once optimization algorithms must balance feasibility against optimality and any such comparison is very much affected by the scaling of the various quantities. For example, the reduced gradient norm is compared to the state constraint residual norm in several different places in the algorithm. Such a comparison is impossible to make with quantities left in their original scaling. Even though they are simpler algorithms, issues of scaling are also critical in several different types of algorithms for unconstrained optimization and nonlinear equations. All of these problems are magnified in constrained optimization algorithms. The theory on scaling and its impact on numerical algorithms is not very solid (see [3]) and scaling remains a necessary, but poorly understood, black art in numerical computing.

7 Conclusions

There are a number of conclusions that we have drawn as a result of this milestone work:

- Predictive simulations capable of answering tomorrows questions mandates moving beyond the basic forward solve and requires the incorporation of invasive technologies for sensitivities, optimization, and other advanced numerical algorithms.
- Solving complex numerical problems (such as transient sensitivities) to the highest quality with the greatest efficiency requires the vertical integration of many different types of advanced numerical algorithms that can be tailored to the specific problem.
- The vertical integration of a large number of advanced numerical algorithms requires the development and adoption of standard interfaces.
- Thyra standard interfaces for linear operators and vectors, preconditioners and linear solvers, nonlinear models, and nonlinear solvers have allowed the vertical integration of a large variety of numerical solvers and access to a variety of nonlinear applications.
- Application codes must present themselves as a ModelEvaluator and then hand over nearly complete control to the numerical solver(s) in order to take full advantage of advanced nonlinear numerical algorithms. Monolithic forward time stepping application codes require significant nodification to take advantage of these more sophisticated solution techniques.
- Nightly building and testing of the development versions of the application and Trilinos:
  - results in better production capabilities and better research,
  - brings algorithm developers and application developers closer together allowing for a better exchange of ideas and concerns,
  - refocuses Trilinos developers on customer efforts,
  - helps drive research-quality algorithm development, and
  - reduces barriers for new algorithms to have impact on production applications.
- Other application projects and scientific support software projects should consider adopting the type of continuous integration that is used with Charon + Trilinos that was developed as part of this milestone work.

References

[1] R. A. Bartlett, B. G. van Bloeman Waanders, and M. A. Heroux. Vector reduction/transformation operators for linear algebra interfaces to efficiently develop complex abstract numerical algorithms independently of data mapping, 2004. *ACM TOMS*.

[2] S. Chandrasekhar. *Hydrodynamic and Hydromagnetic Stability*. Dover, 1981.

[3] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, 1996.

[4] Paul M. Duvall. *Continuous Integration: Improving Software Quality and Reducing Risk*. Addison-Wesley, 2007.

[5] Mike A. Heroux, Teri Barth, David Day, Rob Hoekstra, Rich Lehoucq, Kevin Long, Roger Pawlowski, Ray Tuminaro, and Alan Williams. Trilinos : object-oriented, high-performance parallel solver libraries for the solution of large-scale complex multi-physics engineering and scientific applications. `http://software.sandia.gov/Trilinos`.

[6] Sandia National Laboratories, `http://mpcharon.sandia.gov/`. *Charon Device Simulator*.

[7] Sandia National Laboratories, `https://cfwebprod.sandia.gov/cfdocs/qaspr/templates/index.cfm`. *Qualification Alternatives to SPR*.

[8] Charles F. Van Loan. *Introduction to Scientific Computing*. Prentice Hall, Inc., 1997.

DISTRIBUTION:

2 MS 9018 Central Technical Files, 8944

2 MS 0899 Technical Library, 4536

Sandia National Laboratories