SANDIA REPORT

SAND98–0340 • UC–705
Unlimited Release
Printed February 1998

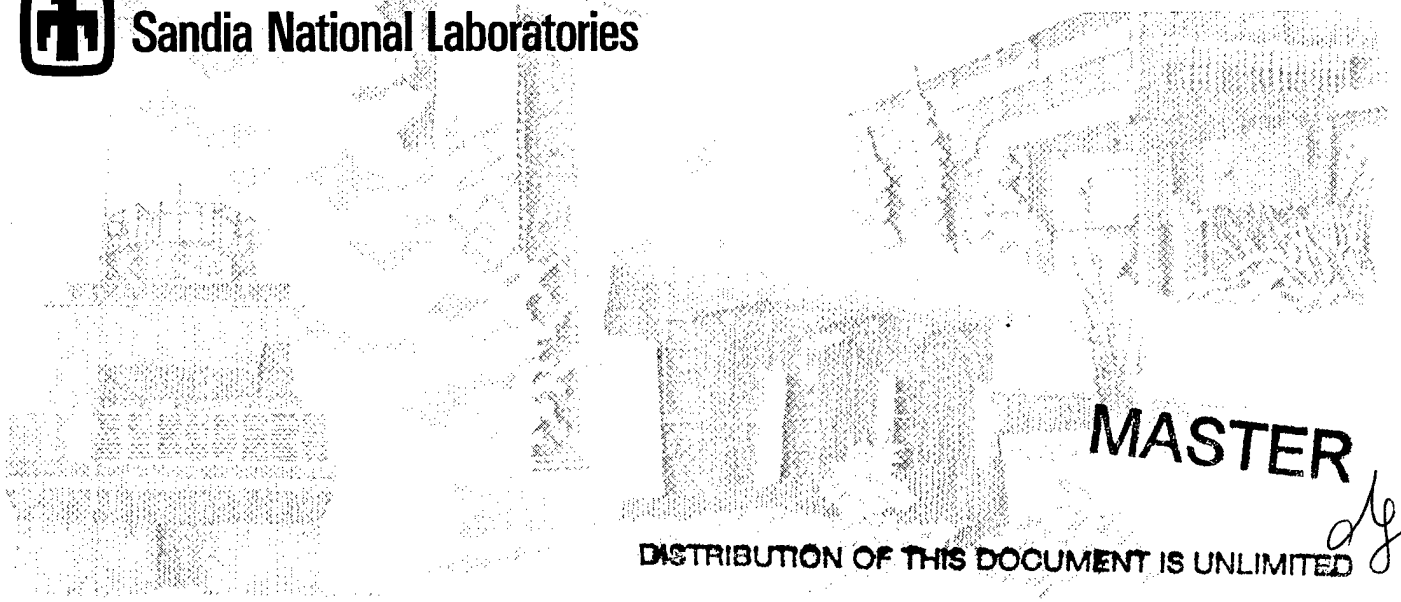
Optimization Strategies for Complex Engineering Applications

Michael S. Eldred

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 under Contract DE-AC04-94AL85000.

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
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

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

NTIS price codes
Printed copy: A06
Microfiche copy: A01

DISCLAIMER

SAND98-0340
Unlimited Release
Printed February 1998

Distribution
Category UC-705

Optimization Strategies for Complex Engineering Applications

Michael S. Eldred
Structural Dynamics Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0439

Abstract

LDRD research activities have focused on increasing the robustness and efficiency of optimization studies for computationally complex engineering problems. Engineering applications can be characterized by extreme computational expense, lack of gradient information, discrete parameters, nonconverging simulations, and nonsmooth, multimodal, and discontinuous response variations. Guided by these challenges, the LDRD research activities have developed application-specific techniques, fundamental optimization algorithms, multilevel hybrid and sequential approximate optimization strategies, parallel processing approaches, and automatic differentiation and adjoint augmentation methods. This report surveys these activities and summarizes the key findings and recommendations.

Acknowledgment

The work of this LDRD involved many technical staff and contractors across Sandia-Albuquerque and Sandia-Livermore. The author would like to recognize Bill Hart (9222) for his contributions in genetic algorithms, pattern search, and asynchronous approaches; Bill Bohnhoff (9341) for his software architecture development and support of research activities and applications; Chris Moen (8345), Juan Meza (8950), and Todd Plantenga (8950) for their work on CVD applications and gradient calculation techniques; Rick Eisler (9622) for his work with ADIFOR; Vicente Romero (9113) for his work on fire surety applications; Walt Witkowski (9234) and Ken Chen (9111) for their work in die design applications; Andy Salinger and Scott Hutchinson (both 9221) for their work in parallel optimization of CVD plant operations; Eric Ponslet (Amparo Corporation) for his work on discrete optimization of vibration isolation systems; David Zimmerman (University of Houston) for his work in neural networks and sequential approximate optimization with genetic algorithms; Tom Paez (9741) for his work in evaluation of different neural network formulations; and David Outka (2526) for his vision and numerous contributions during the initial stages of the LDRD. The details of these contributions are shown in the referenced conference papers, journal papers, memoranda, and SAND reports.

The author would also like to thank Chris Moen and Bill Hart for their helpful comments during the review of this manuscript.

Contents

Preface

This document is a final report for the "Optimization Strategies for Computationally Challenging Problems in Engineering Sciences" LDRD project which was funded for fiscal years 1995 through 1997. It summarizes the main discoveries and research contributions of this LDRD activity and includes many of the relevant conference and journal papers that arose from the activity in the appendices.

The development of the DAKOTA iterator toolkit software can be attributed in part to support from this LDRD. It is documented in a separate companion report.

Optimization Strategies for Complex Engineering Applications

Introduction

Computational methods developed for fluid mechanics, structural dynamics, heat transfer, nonlinear structural mechanics, shock physics, and numerous other fields of engineering can be an enormous aid to understanding the complex physical systems they simulate. Often, it is desired to use these simulations as virtual prototypes to obtain an acceptable or optimized design for a particular system. This project seeks to enhance the utility of computational methods by enabling their use as design tools, so that simulations may be used not just for single-point predictions, but also for automated determination of system performance improvements throughout the product life cycle. System performance objectives, to name a few possibilities, can be formulated to minimize weight, cost, or defects; to limit a critical temperature, stress, or vibration response; or to maximize performance, reliability, throughput, reconfigurability, agility, or design robustness. A systematic, rapid method of determining these optimal solutions leads to better designs and improved system performance, reduces dependence on prototypes and testing, and shortens the design cycle and reduces development costs.

Toward these ends, a general purpose toolkit has been developed for the integration of commercial and in-house analysis capabilities with broad classes of systems analysis tools. Written in C++, the DAKOTA (Design Analysis Kit for OpTimizAtion) toolkit is a flexible, extensible interface between analysis codes and iteration methods. In addition to optimization methods and strategies, the DAKOTA toolkit implements uncertainty quantification with direct and sampling methods, parameter estimation with nonlinear least squares solution methods, and sensitivity analysis with general-purpose parameter study capabilities. By employing object-oriented design to implement abstractions of the key concepts involved in iterative systems analyses, the DAKOTA toolkit provides a flexible and extensible problem-solving environment for current and future problems of interest. Through DAKOTA, point solutions from simulation codes can be used for answering more fundamental engineering questions, such as "what is the best design?", "how safe is it?", or "how much confidence do I have in my answer?".

In addition to its role as a problem-solving environment, the DAKOTA toolkit also provides a platform for research and development of advanced methodologies which focus on increasing the robustness and efficiency of systems analyses for computationally complex engineering problems. These methodologies have been the primary focus of the "Optimization Strategies for Computationally Challenging Problems in Engineering Sciences" LDRD and are the main subject of this report. The specifics of the DAKOTA software are provided in a separate report[1].

Guided by the challenges observed from optimizing complex engineering applications, the LDRD research activities have focused on developing fundamental algorithms, multilevel hybrid and sequential approximate optimization strategies, parallel processing approaches, and automatic differentiation and adjoint augmentation methods. The following sections summarize the application challenges that were observed and how the cited development areas addressed these challenges.

Complex Engineering Applications

Application investigations[2-7] have shown the challenges involved in interfacing optimization methods with complex engineering simulations. Reference 2 provides a summary of the following applications: shape optimization of a hazardous material transportation cask (see also References 8, 9, and 10), determination of worst case fire environments (see also References 11 and 12), coating flow die design (see also Reference 13), and discrete optimization of a vibration isolation platform (see also Reference 14). Reference 3 extends the worst case fire environment studies as well as adding an application in chemical vapor deposition (CVD) plant design using MP chemically reacting flow simulations. References 4 and 5 document various facets of an application in heat transfer design of CVD reactors, and References 6 and 7 document a pair of parameter estimation applications performed with DAKOTA by its growing user community. References 2 and 3 are included as Appendices A and B.

The following list (originally presented in Reference 2) presents several common challenges associated with complex engineering simulations.

1. The time required to complete a single function evaluation with one parameter set is large. Hence, minimization of the number of function evaluations is vital.

2. Analytic derivatives (with respect to the parameters) of the objective and constraint functions are frequently unavailable. Hence, sensitivity-based optimization methods depend upon numerically generated gradients which require additional function evaluations for each scalar parameter.

3. The parameters may be either continuous or discrete, or a combination of the two.

4. The objective and constraint functions may not be smooth or well-behaved; i.e., the response surfaces can be severely nonlinear, discontinuous, or even undefined in some regions of the parameter space. The existence of several local extrema (multimodality) is common.

5. Convergence tolerances in embedded iteration schemes introduce nonsmoothness (noise) in the function evaluation response surface, which can result in inaccurate numerical gradients.

6. Each function evaluation may require an "initial guess." Function evaluation dependence on the initial guess can cause additional nonsmoothness in the response surface. Moreover, a solution may not be attainable for an inadequate initial guess, which can restrict the size of the allowable parameter changes.

Each of the listed challenges can adversely affect the robustness and efficiency of optimization processes and has led to research and development of application-specific techniques, new optimization algorithms, multilevel hybrid and sequential approximate optimization strategies, parallel optimization approaches, and automatic differentiation and adjoint augmentation methods. The following sections describe these techniques and how they mitigate the cited difficulties.

Techniques for Application-Specific Challenges

This section contains application-specific techniques which can be used to address the challenges associated with complex engineering applications. While they are not universally applicable to all situations, they can be highly effective in certain instances.

Adaptive simulation termination was employed in the worst-case fire and cask shape optimization applications[2], both to minimize computational expense and to guarantee event capture. Since the duration of transient simulations necessary to capture the events of interest (subcomponent failures and maximum stress during impact, respectively) was a function of the design variables (fire boundary conditions and impact and thermal layer thicknesses, respectively), it was necessary to monitor simulation progress and terminate execution when event capture was determined. In the worst-case fire application, event capture was signaled when critical node temperatures exceeded failure thresholds, and in the cask application, crush event capture was indicated by rebound in the kinetic energy histories. These techniques were highly effective in conserving compute cycles while guaranteeing that the simulations did not terminate prematurely and truncate the responses.

It is not uncommon for a complex simulation to fail to obtain a converged solution for a particular parameter set. In some cases, this results from an inadequate initial guess, for which simulation continuation methods can be highly effective. Continuation can be implemented within the analysis code (e.g., MPSalsa, GOMA) or externally to the code (e.g., DAKOTA's continuation option for simulation failure capturing). In other cases, the solution may be transitioning into a physics regime which the simulation is incapable of resolving (e.g., turbulence, phase change, structural instability) regardless of initial guess; in these cases, continuation will be ineffective. Assuming that this regime is undesirable from a design perspective (which should be carefully evaluated), the existing simulation code can still be utilized successfully if the optimization can be configured to recover from these parameter sets. One simple approach for discouraging optimization algorithms from entering the unsolvable regime is to return an artificially poor objective function (or, for a constrained problem, a stability constraint violation) in place of the nonconverging simulation results. This is DAKOTA's recovery option for simulation failure capturing. The difficult part of the recovery process can be the detection of the failure to converge, since some analysis codes may loop indefinitely, or worse, return erroneous results after a fixed amount of effort. A more sophisticated technique is to build a strategy around an algorithm that decreases its extent of optimization (e.g., trust region radius, linear programming move limits, pattern search delta, etc.) when analysis failure is detected. The motivation for the more sophisticated approach would be the ability to implement increased flexibility beyond the internal optimizer mechanisms relied on in the recovery approach.

Even when you can obtain a solution, the variation of the solution responses with the parameters may be poorly behaved. In particular, it may be noisy and nonsmooth. This behavior may arise from any number of factors, including under-resolved meshing, inadequate time step and solver convergence controls, improper physics models, modeling errors, insufficient precision, distorted elements or high element aspect ratios, etc. Some of these problems are errors that must be corrected from a modeling perspective before meaningful optimization can be performed (see, for example, element aspect ratios and contact line singularities in the cask shape optimization application[2]). Other problems are tightly linked with the individual computational expense of the simulations (such as mesh density, time step size, solver convergence tolerance - see cask, worst-case fire, and die design applications[2]) and must be managed so that the modeling fidelity is sufficient to allow optimizer navigation but not so over-resolved as to be computationally wasteful. "Sufficient" modeling fidelity is tightly linked to the optimizer in use, which can be cleverly exploited in multiple method approaches which employ variable complexity modeling (see Reference 3 and Multilevel Hybrid Optimization discussion below). Also, global approximations (e.g., response surfaces, neural networks) can significantly lessen the correlation between simulation expense and optimizer navigability by placing a differentiable intermediary between the optimizer and the actual simulations in the context of sequential approximate optimization (see discussion below). Unfortunately, many of the expense versus navigability trade-offs are most evident in hindsight and can be difficult to identify *a priori* in new applications with little experience base. Parameter study investigations of design space variations on a fine scale can be extremely helpful in identifying smoothness properties in these cases.

While local nonsmoothness is generally an artifact of incompletely resolved physics, the challenges of multimodality, discontinuity, and nonconvexity in general may be real features of an application which improvements in model fidelity will not affect. In some cases, changes in problem formulation may help, but in most cases, the optimization approach must be carefully selected to address these challenges. Traditional gradient-based methods assume continuous, convex, unimodal domains. Newton-based approaches additionally assume a quadratic model. In the worst case fire application[2,3], a problem with multiple minima, nonsmoothness, and slope discontinuities, it was observed that Newton-based methods performed very poorly, which is not surprising since the cusp at the minimum is poorly captured by a quadratic model. First-order methods (which do not retain curvature information - e.g., conjugate gradient) were much more successful, even though they are generally regarded as inferior in their convergence properties to Newton-based approaches. Zero-order methods (such as coordinate pattern search) were the most robust in reliably locating the local minimum closest to the starting point. Lastly, the multimodality of the problem had to be addressed with the use of global approaches, such as genetic algorithms, local search from multiple starting points, or stratified Monte Carlo sampling. These observations should not be interpreted as reasons to avoid sophisticated gradient-based approaches, since they can be significantly more efficient on smooth, convex domains. Rather, the important conclusion is that the fewer assumptions a method makes about its domain, the more robust it tends to be when its domain is not continuous, convex, or unimodal. Furthermore, the more highly tuned an approach is for maximum efficiency on convex domains, the worse it may perform on nonconvex domains. Parameter

study investigations of design space variations on a coarse scale can be extremely helpful in identifying these challenges.

With many of the above challenges, a compounding effect can be the fact that an optimizer will exploit weaknesses in the model if they result in improvements to the objective or constraints. Put simply, optimizers are very adept at breaking models. The model must not only be valid for the nominal parameters, it must also be robust with respect to parameter changes. For example, analysis truncation caused by insufficient simulation duration is very likely to cause erroneous reductions in the "peak" response information returned to the optimizer. If this response is being minimized or constrained from above, the optimizer will naturally navigate in the direction of parameter sets which cause the event of interest to take even longer to complete, since this will result in more truncation and lower peak response. The optimizer's only feedback from the simulation is the objective and constraints, so it is unaware of the fact that the reductions are not real, but rather artifacts of model breakdown. Another example is that of entering a region of the design space which the simulation cannot resolve properly or for which the model is inadequate. The responses returned will be erroneous, and their inaccuracy may again be viewed as beneficial by the optimizer depending on the design problem formulation. Thus, the user must be mindful that modeling problems can be particularly insidious in that they can be exploited in the pursuit of design objectives. If they are not prevented or at least detected, then the situation can quickly deteriorate to one of "garbage in, garbage out."

Fundamental Algorithm Development

Activities involved in the development of fundamental optimization algorithms have focused on extensions to the OPT++[15, 16] and SGOPT[17] software libraries. OPT++ has added bound constrained and barrier function extensions to its Newton-based optimizers and has added Gauss-Newton and ellipsoid methods. SGOPT has added genetic algorithm/local search hybrids, evolutionary pattern search methods, and a variety of coordinate pattern search techniques, as well as asynchronous capabilities for parallel genetic algorithms and parallel coordinate pattern search. In addition, theoretical work in convergence proofs for evolutionary pattern search and stopping rules for random sampling has been performed. SGOPT's parallel algorithms were employed to demonstrate single-level parallel optimization with the worst-case fire application (see Reference 3 and parallel optimization discussion below).

When variables are discrete, the concept of a derivative breaks down and recourse must be taken to nongradient-based combinatorial methods. Discrete and mixed continuous-discrete problems can be tackled with genetic algorithms, branch and bound, simulated annealing, and many other nongradient-based approaches. For example, genetic algorithms were employed in the discrete problem of optimal isolator placement for the vibration isolation testbed[2]. Discrete and mixed continuous-discrete problem domains are a continuing focus of the SGOPT development activity[17].

Multilevel Hybrid Optimization

The fact that response variation smoothness and simulation expense are often correlated through simulation controls (e.g., simulation time step size, simulation convergence tolerance) can be exploited by employing hybrid algorithms and multiple models of varying fidelity. These hybrids are described as "multilevel" to denote the fact that there is clear separation between the iterators within the hybrid strategy. Rather than combining the traits of different methods into a composite algorithm, multilevel hybrids execute one iterator and then pass its best results to the next iterator which then refines those results.

By using inexpensive and nonsmooth models with nongradient-based methods and expensive and smooth models with gradient methods, it was shown that coordinate pattern search/nonlinear programming hybrids were more efficient than nonlinear programming alone and more accurate than coordinate pattern search alone[3]. This validated the concept of multilevel hybridization and emphasized the importance of matching the required model fidelity to the iterator in use. DAKOTA has subsequently been extended to manage the binding of multiple models with multiple iterators by embedding the concepts of variable complexity modeling within its multilevel hybrid strategy (this important software development also enables other multiple model, multiple iterator strategies). Continuing work in hybridization is focusing on development and implementation of (1) fine-grained control of multilevel hybrids in which performance metrics are computed on each cycle and used to govern switch/adapt logic, and (2) more tightly-coupled hybrids in which local searches are used to improve selected points within a global search.

Sequential Approximate Optimization

Sequential approximate optimization (SAO) involves the sequential use of approximations in seeking to converge to an optimal solution while minimizing the number of actual function evaluations that have to be performed. In SAO's simplest form, an approximation is built, an optimizer computes an optimal solution over the current approximation, this optimal point is evaluated with a "truth" model and used to update the approximation, and the cycle continues until convergence. The DAKOTA capability for multiple models and iterators enables the presence of approximate and truth models in the SAO strategy, and the "DakotaInterface" class hierarchy provides a flexible and extensible framework for implementation of approximation techniques[1]. Candidate approximation techniques have been surveyed[18] and evaluated for applicability within the DAKOTA framework. Response surface and neural network techniques, using design and analysis of computer experiments (DACE) preprocessors, and multipoint approximations were recommended as general-purpose approaches based on their application independence. Reference 18 is included as Appendix C.

Research in direct training of artificial neural networks has been performed in seeking to identify promising techniques for design space approximation which are well-behaved when updated with promising points from SAO cycles[19]. This capability was used in a demonstration of SAO using genetic algorithms (GAs) in which both fitness and diversity were used as metrics to select the best GA individuals to be used for sequentially updating

a neural network approximation. Direct training of neural networks has also been compared with more widely-used back-propagation approaches[20]. While direct training is more efficient for many problems of interest, it can be less accurate than back-propagation for the same number of hidden layer neurons. In addition, the ability to adapt to a changing input/output functional relationship (e.g., variable complexity modeling) favors back-propagation over direct training.

In the basic SAO approach presented above, there are no guarantees of convergence and a poor approximation may not lead to improvement in the actual objective function. The new field of model management frameworks[21,22] promises to augment heuristic SAO approaches with robust, provably convergent backplanes which can be efficient when the approximations are accurate and robust when they are not.

Parallel Optimization

Single-level parallel optimization approaches, in which either multiple single-processor simulations execute simultaneously (parallelism in the optimization algorithm) or a single simulation executes in parallel (parallelism in the simulation), have been investigated[3]. In the worst-case fire application, multiple simultaneous executions of single-processor heat transfer simulations were employed with parallel coordinate pattern search to reduce wall clock time by up to a factor of 20 over the benchmark nonlinear programming approach using up to six nodes of an IBM SP2. In this case, the optimization approach was parallel, although the simulations were not. In the CVD plant design application, the MPSalsa massively parallel chemically reacting flow code[23,24] was used on the Intel Paragon as a function evaluator within a gradient-based optimization approach. Here, the simulation was parallel (executing on either 256 or 512 processors), but the optimization approach was not.

Both of these single-level parallel approaches, while effective, have clear limitations. In the case of the worst-case fire application, speedup was limited by the number of independent simulations to be performed (two times the number of design variables for coordinate pattern search). In the CVD plant application, speedup was limited by the practical limit on the number of processors for a given problem size (as processors increase for a given problem size, communication begins to dominate computation and parallel effectiveness decreases). Combining the two single-level approaches into multilevel parallelism, in which multiple MP simulations execute simultaneously, extends the achievable speedup far beyond either single-level approach. Additional levels of parallelism beyond the two cited (e.g., multiple independent genetic algorithm populations evolving simultaneously) also exist and can be used to further extend machine utilization possibilities. Multilevel parallelism is currently being pursued on the Intel TeraFLOPS with funding from the Accelerated Strategic Computing Initiative (ASCI)[25].

Gradient Calculation Techniques

When analytic gradients are unavailable from an analysis code, several approaches can be pursued. One can manually extend the analysis source code to additionally compute

gradient information using either the direct or adjoint approaches, employ automatic differentiation (AD) to augment the analysis capability, use finite differences, employ nongradient-based optimization approaches, or build differentiable approximate surfaces from point-solution data.

Manually extending the analysis is the most effort-intensive approach, but it can be the most accurate and computationally efficient. One can make use of the latest research developments in sensitivity analysis[26,27] to compute gradients. In an electronics packaging application[28], the adjoint approach was shown to reduce time to convergence by a factor of two over finite difference approaches. In general, the direct approach is preferred if the number of design variables is smaller than the number of responses to be differentiated, and the adjoint approach is preferred if the number of responses is smaller than the number of design variables. The AD approach uses compiler technology to chain rule response computations. The effort required can be low since it is mostly automatic, but it can result in very large executables with significantly longer run times than the original analysis. While generally more accurate than finite differencing, relative efficiency depends on the simulation code, the responses being differentiated, and the size of the optimization problem. In a CVD reactor application[29], AD was shown to generate considerably more accurate gradients than finite differencing and was less expensive than finite differencing when the number of design parameters exceeded 4 or 16, for the TWAFER and TACO codes respectively. In an application seeking maximum impact velocity for a guided projectile, AD was shown to improve robustness by bypassing finite differencing pitfalls and reduce the time to convergence by a factor of seven[30]. References 28 and 29 are included as Appendices D and E. Finite differences can compound expense on multiple levels both by requiring additional function evaluations and by requiring response variations which are smooth on the scale of the finite difference step. Nongradient-based optimization approaches can be effective, especially for challenging domains, although the number of simulation executions required will usually be higher than for gradient-based approaches. This can often be offset by relaxation of smoothness requirements with its associated reductions in individual function evaluation expense. Lastly, global approximations (e.g., response surfaces) can be used to build differentiable surfaces from point-solution data.

Each of these approaches can be effective, and it is often considerations of the up-front human effort required to achieve maximum efficiency which govern the selection. If simulation codes will be used frequently for design studies, then extension of the codes through manual modification or automatic differentiation may pay large dividends in improved optimization efficiency. However, in cases of limited analysis code usage, use of finite differencing or nongradient-based approaches can minimize human set-up time.

Conclusions

Engineering applications at Sandia National Laboratories share many challenging features, most notably extreme computational expense and nonsmoothness and nonconvexity of response variations. If these simulations are to be used as virtual prototypes to improve products and processes, then these challenges must be addressed with robust and efficient

optimization methods and strategies. This LDRD has performed research investigations in advanced optimization methodologies and has quantified the performance of these approaches on complex engineering applications in a wide range of engineering disciplines.

In particular, adaptive simulation termination, simulation failure capturing through continuation and/or recovery, nonsmoothness management through variable fidelity modeling and approximations, and management of nonconvexity through global and robust local algorithms are effective application-specific techniques which can manage simulation expense while providing for successful optimizer navigation. Novel algorithms being developed in the OPT++ and SGOPT libraries are providing new capabilities specifically designed for complex engineering applications. Multilevel hybrid strategies are employing multiple iterators with multiple models of varying fidelity to exploit the strengths of different methods in identifying optimal solutions quickly and reliably. Sequential approximate optimization strategies are employing design space approximations to inexpensively compute improvements in the actual objective function. Single-level parallel approaches have shown considerable promise and have highlighted the need for multilevel parallelism for achieving peak computational speed. And, automatic differentiation and adjoint methods for computing gradients have provided highly effective replacements to the expensive process of finite differencing.

This collection of findings culminates in (1) an improved understanding of the challenges involved in optimizing with complex leading-edge simulations and (2) a suite of advanced capabilities, many implemented in the DAKOTA software, which can address these challenges and deliver optimal solutions reliably and expediently.

References

[1]Eldred, M.S, Bohnhoff, W.J., and Hart, W.E., "DAKOTA, An Object-Oriented Framework for Design Optimization, Parameter Estimation, Sensitivity Analysis, and Uncertainty Quantification," Sandia Technical Report SAND98-XXXX, In preparation.

[2]Eldred, M.S., Outka, D.E., Bohnhoff, W.J., Witkowski, W.R., Romero, V.J., Ponslet, E.R., and Chen, K.S., "Optimization of Complex Mechanics Simulations with Object-Oriented Software Design," *Computer Modeling and Simulation in Engineering*, Vol. 1, No. 3, August 1996. Revised and extended from Eldred, M.S., Outka, D.E., Fulcher, C.W., and Bohnhoff, W.J., "Optimization of Complex Mechanics Simulations with Object-Oriented Software Design," paper AIAA-95-1433 in *Proceedings of the 36th AIAA/ASME/ASCE/AHE/ASC Structures, Structural Dynamics, and Materials Conference*, New Orleans, LA, April 10-13, 1995, pp. 2406-2415. Included as Appendix A.

[3]Eldred, M.S., Hart, W.E., Bohnhoff, W.J., Romero, V.J., Hutchinson, S.A., and Salinger, A.G., "Utilizing Object-Oriented Design to Build Advanced Optimization Strategies with Generic Implementation," paper AIAA-96-4164 in *Proceedings of the 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, Sept. 4-6, 1996, pp. 1568-1582. Included as Appendix B.

[4]Meza, J.C., and Plantenga, T.D., "Optimal Control of a CVD Reactor for Prescribed Temperature Behavior," Sandia Technical Report SAND95-8224, April 1995.

[5]Moen, C.D., Spence, P.A., and Meza, J.C., "Optimal Heat Transfer Design of Chemical Vapor Deposition Reactors," Sandia Technical Report SAND95-8223, April 1995.

[6]Blackwell, B.F., and Eldred, M.S., "Application of Reusable Interface Technology for Thermal Parameter Estimation," *Proceedings of the 1997 National Heat Transfer Conference*, session on Inverse Design Problems in Heat Transfer and Fluid Flow, August 1997.

[7]Hobbs, M. L., "A Global HMX Decomposition Model," 1996 JANNAF Propulsion Systems Hazards Subcommittee Meeting, Naval Postgraduate School, Monterey, CA, Nov. 4-8, 1996.

[8]Harding, D.C., Eldred, M.S., and Witkowski, W.R., "Integration of Finite Element Analysis and Numerical Optimization Techniques for RAM Transport Package Design," *Proceedings of the 11th International Conference on the Packaging and Transportation of Radioactive Materials* (PATRAM `95), Las Vegas, NV, Dec. 3-8, 1995.

[9]Harding, D.C., and Eldred, M.S., "Radioactive Material Transportation Package Design Using Numerical Optimization Techniques," *Proceedings of the 1995 Joint ASME/ JSME Pressure Vessels and Piping Conference*, Honolulu, Hawaii, July 23-27, 1995, Vol. PVP-307, pp. 29-36.

[10]Witkowski, W.R., Eldred, M.S., and Harding, D.C., "Integration of Numerical Analysis Tools for Automated Numerical Optimization of a Transportation Package Design," *Proceedings of the 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, paper AIAA94-4259, Panama City Beach, FL, Sept. 7-9, 1994.

[11]Romero, V.J., Painton, L.A., and Eldred, M.S., "Optimization Under Uncertainty: Shifting of Maximum Vulnerability Point Due to Uncertain Failure Thresholds," 1997 INFORMS Spring Meeting, San Diego, CA, May 1997.

[12]Romero, V.J., Eldred, M.S., Bohnhoff, W.J., and Outka, D.E., "Application of Optimization to the Inverse Problem of Finding the Worst-Case Heating Configuration in a Fire," *Proceedings of the 9th International Conference on Numerical Methods in Thermal Problems*, Altlanta, GA, July 17-21, 1995, Vol. 9, Part 2, pp. 1022-1033.

[13]Chen, K.S., and Witkowski, W.R., "Design Optimization of Liquid-Distribution Chamber-Slot Dies Using the DAKOTA Toolkit," 50th Annual Conference of the Society for Imaging Science and Technology, Cambridge MA, May 18-23, 1997.

[14]Ponslet, E.R., and Eldred, M.S., "Discrete Optimization of Isolator Locations for Vibration Isolation Systems: an Analytical and Experimental Investigation," paper AIAA-96-4178 in *Proceedings of the 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, Sept. 4-6, 1996, pp. 1703-1716. Also appears as Sandia Technical Report SAND96-1169, May 1996.

[15]Meza, J.C., "OPT++: An Object-Oriented Class Library for Nonlinear Optimization," Sandia Technical Report SAND94-8225, March 1994.

[16]Meza, J.C., Moen, C.D., Plantenga, T.D., Spence, P.A., Tong, C.H., Hendrickson, B.A., Leland, R.W., Reese, G.M., "Parallel optimization methods for agile manufacturing," Sandia Technical Report SAND97-8275, August 1997.

[17]Hart, W.E., "SGOPT, A C++ Library of (Stochastic) Global Optimization Algorithms," Sandia Technical Report SAND98-XXXX, In preparation.

[18]Eldred, M.S., "Survey of Approximation Methods in Optimization," memo to distribution, July 28, 1995. Included as Appendix C.

[19]Zimmerman, D.C., "Genetic Algorithms for Navigating Expensive and Complex Design Spaces," Final Report for Sandia National Laboratories contract AO-7736 CA 02, Sept. 1996.

[20]Paez, T., "Assessment of Artificial Neural Network," memo to M. Eldred, November 20, 1997.

[21]Serafini, D., "Software for Managing Approximate Models in Multidisciplinary Optimization," paper AIAA-96-4104 in *Proceedings of the 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, Sept. 4-6, 1996, pp. 1063-1067.

[22]Alexandrov, N., "Robustness Properties of a Trust Region Framework for Managing Approximations in Engineering Optimization," paper AIAA-96-4102 in *Proceedings of the 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, Sept. 4-6, 1996, pp. 1056-1059.

[23]Shadid, J.N., Moffat, H.K., Hutchinson, S.A., Hennigan, G.L., Devine, K.D., and Salinger, A.G., *MPSalsa - A finite element computer program for reacting flow problems. Part 1 - Theoretical development.* Sandia Technical Report SAND95-2752, Albuquerque, NM, 1996.

[24]Salinger, A.G., Devine, K.D., Hennigan, G.L., Moffat, H.K., Hutchinson, S.A., and Shadid, J.N., *MPSalsa - A finite element computer program for reacting flow problems. Part 2 - User's Guide*, Sandia Technical Report SAND96-2331, 1996.

[25]Eldred, M.S., Hart, W.E., Bohnhoff, W.J., and Rhea, R.E., "Design and Implementation of Multilevel Parallel Optimization on the Intel TeraFLOPS," submitted to *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, Sept. 2-4, 1998.

[26]Tortorelli, D.A., and Michaleris, P., "Design Sensitivity Analysis: Overview and Review," *Inverse Problems in Engineering*, Vol. 1, 1994, pp. 71-105.

[27]Adelman, H.A., and Haftka, R.T., "Sensitivity Analysis of Discrete Systems," in *Structural Optimization: Status and Promise*, Kamat, M.P., Ed., Progress in Astronautics and Aeronautics, Volume 150, American Institute of Aeronautics and Astronautics, Washington, DC, 1993, pp. 291-315.

[28]Moen, C.D., "The Discrete Adjoint Method for Gradient Evaluation in Optimization," unpublished. Included as Appendix D.

[29]Moen, C.D., Spence, P.A., Meza, J.C., and Plantenga, T.D., "Automatic Differentiation for Gradient-Based Optimization of Radiatively Heated Microelectronics Manufacturing Equipment," paper AIAA-96-4118 in *Proceedings of the 6th AIAA/USAF/NASA/ ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, Sept. 4-6, 1996, pp. 1167-1175. Included as Appendix E.

[30]Eisler, R., "An experience with ADIFOR in computing analytical Jacobians (when you desperately want to avoid finite differences)," memo to M. S. Eldred, August 26, 1996.

APPENDIX A

Optimization of Complex Mechanics Simulations with Object-Oriented Software Design

Optimization of Complex Mechanics Simulations with Object-Oriented Software Design*

M.S. Eldred†, D.E. Outka‡, W.J. Bohnhoff§, W.R. Witkowski¶,
V.J. Romero#, E.R. Ponslet**, and K.S. Chen††

Sandia National Laboratories‡‡
Albuquerque, NM 87185

Abstract

The benefits of applying optimization to computational models are well known, but their range of widespread application to date has been limited. This work attempts to extend the disciplinary areas to which optimization algorithms may be readily applied through the development and application of advanced optimization strategies capable of handling the computational difficulties associated with complex simulation codes. Towards this goal, a flexible software framework is under continued development for the application of optimization techniques (and other iterative computational methods) to broad classes of engineering applications, including those with high computational expense and nonsmooth, nonconvex design space features. Object-oriented software design with C++ has been adopted as a tool to provide a flexible, extensible, and robust multidisciplinary toolkit that establishes the protocol for wrapping parameter optimization around computationally-intensive simulations. The object-oriented approach is well-suited for handling this large software undertaking, in which a wide assortment of optimization algorithms, approximation techniques, and hybridized strategies must be generically interfaced with broad classes of analysis capabilities. Demonstrations of optimization using the software are presented in fluid mechanics, heat transfer, nonlinear solid mechanics, and structural dynamics. Optimal results are presented along with technical lessons that were learned in the optimization process.

Introduction

Computational methods developed in fluid mechanics, structural dynamics, heat transfer, nonlinear large-deformation mechanics, manufacturing and material processes, and many other fields of engineering can be an enormous aid to understanding the complex physical systems they simulate. Often, it is desired to utilize these simulations as virtual prototypes to improve or optimize the design of a particular system. This effort seeks to enhance the utility of this broad class of computational methods by providing them with a general optimization capability and enabling their use as design tools, so that simulations may be used not just for single-point predictions, but also for improving system performance in an automated fashion. System performance objectives can be formulated to minimize weight or defects or to maximize performance, reliability, throughput, reconfigurability, agility, or design robustness (insensitivity to off-nominal parameter values). A systematic, rapid method of determining these optimal solutions will lead to better designs and improved system performance and will reduce dependence on prototypes and testing, which will shorten the design cycle and reduce development costs.

*Presented as paper AIAA-95-1433-CP at the 36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, New Orleans, LA, April 10-13, 1995.

†Senior Member of Technical Staff, Structural Dynamics Department, Mail Stop 0439.

‡Senior Member of Technical Staff, Navigation Guidance and Control System Department, Mail Stop 1174.

§Senior Member of Technical Staff, Intelligent Systems Department, Mail Stop 1177.

¶Senior Member of Technical Staff, Structural Dynamics Department, Mail Stop 0439.

#Senior Member of Technical Staff, Thermal Sciences Department, Mail Stop 0835.

**Amparo contractor, Structural Dynamics Department, Mail Stop 0439.

††Senior Member of Technical Staff, Incompressible Fluid Dynamics Department, Mail Stop 0826.

‡‡P.O. Box 5800, Albuquerque, NM 87185, USA. This work performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract DE-AC04-94AL85000.

Towards these ends, we have targeted the needs of a broad class of computational methods in order to provide a general optimization capability. Much work to date in the optimization community has focused on applying either gradient-based techniques to smooth, convex, potentially expensive problems (e.g., (Kamat 1993)) or global techniques to nonconvex but inexpensive problems (e.g., (Törn and Zilinskas 1989)). When the difficulties of high computational expense and nonsmooth, nonconvex design spaces are coupled together, advanced strategies are required. Moreover, since the challenges of each application are frequently very different, generality and flexibility of the advanced strategies are key concerns. The following list itemizes the primary challenges.

Technical Issues. The coupling of optimization with complex computational methods is difficult, and optimization algorithms often fail to converge efficiently, if at all. The difficulties arise from the following traits, shared by many computational methods:

1. The time required to complete a single function evaluation with one parameter set is large. Hence, minimization of the number of function evaluations is vital.
2. Analytic derivatives (with respect to the parameters) of the objective and constraint functions are frequently unavailable. Hence, sensitivity-based optimization methods depend upon numerically generated gradients which require additional function evaluations for each scalar parameter.
3. Convergence tolerances in embedded iteration schemes introduce uncertainty (noise) in the function evaluation response surface, which can result in inaccurate numerical gradients.
4. The parameters may be either continuous or discrete, or a combination of the two.
5. The objective and constraint functions may not be smooth or well-behaved; i.e., the response surfaces can be severely nonlinear, discontinuous, or even undefined in some regions of the parameter space. The existence of several local extrema (multi-modality) is common.
6. Each function evaluation may require an "initial guess." Function evaluation dependence on the initial guess can cause additional uncertainty in the response surface. Moreover, a solution may not be attainable for an inadequate initial guess, which can restrict the size of the allowable parameter changes.

Addressing these challenges with robust and efficient optimization strategies extends the range of applications where the benefits of optimal solutions can be realized.

Technical Approach. To be effective in addressing these technical issues, one must minimize the computational expense associated with repeated function evaluations (efficiency) and maximize the likelihood of successful navigation to the desired optimum (robustness). The key technology developments needed to achieve these goals are fundamental algorithm research, hybrid optimization algorithms, function approximation strategies, parallel processing, and automatic differentiation. Research in hybridization, approximation, and parallel processing is detailed in a separate paper (Eldred et al. 1996).

In this paper, the software infrastructure design and demonstrations of its use in four engineering mechanics applications will be presented. The generation of optimal solutions for the four applications involves mating existing, stand-alone optimizers (nonlinear programming, genetic algorithms, pattern search) with one or more engineering simulations. Thus, the focus of this paper is on 1) how generic interfacing of iteration with simulation is performed, 2) what application-specific techniques are useful in enabling reliable, efficient optimization studies, and 3) how existing techniques perform and what their weaknesses are when interfacing with complex engineering simulations. The results computed serve as benchmarks for comparison of advanced strategy performance (Eldred et al. 1996), and the lessons learned have helped direct the current research focus areas.

Software Design

The DAKOTA (Design Analysis Kit for OpTimizAtion) toolkit utilizes object-oriented design with C++ (Stroustrup 1991) to achieve a flexible, extensible interface between analysis codes and iteration methods. The scope of iteration methods which may be included in the system currently includes optimization, nondeterministic simulation, and parameter study methods. Likewise, there is breadth in the analysis codes which may be interfaced. Currently, simulator programs in the disciplines of nonlinear solid mechanics, structural dynamics, fluid mechanics, and heat transfer have been accessed. The system also provides a platform for research and development of advanced iteration strategies.

Accomplishing the interface between analysis codes and iteration methods in a sufficiently general manner poses a difficult software design problem. These conceptual design issues are being resolved through the use of object-

oriented programming techniques. In mating an iteration method with an analysis code, generic interfaces have been built such that the individual specifics of each iterator and each analysis code are hidden. In this way, different iterator methods may be easily interchanged and different simulator programs may be quickly substituted without affecting the internal operation of the software. This isolation of complexity through the development of generic interfaces is a cornerstone of object-oriented design (the concept of "one interface, many methods"), and is required for the desired generality and flexibility of the advanced strategy developments (e.g., hybridization, function approximation).

The Application Interface (Figure 1) isolates application specifics from an iterator method. By providing a generic interface for the mapping of a set of parameters (e.g., the vector of design variables "OptParameter") into a set of responses (e.g., an objective function, constraints, and sensitivities in "OptResponse"), the specific complexities of a given problem are hidden from the iterator method. Housed within the Application Interface are three pieces of software. The input filter abstraction ("IFilter") provides a communication link which transforms the set of input parameters (OptParameter) into input files for the simulator program. The simulator program reads the input files and generates results in the form of output files or databases (a driver program/script is optional and is used to accomplish nontrivial command syntax and/or progress monitoring for adaptive simulation strategies). Finally, the output filter abstraction ("OFilter") provides another communication link through the recovery of data from the output files and the computation of the desired response data set (OptResponse). The following descriptions identify the actual C++ classes used by DAKOTA:

Optimizer: This class represents the optimization technique to be used. Many optimizers may be derived from this class, enabling easy selection of a particular optimizer as a problem may require. The **Optimizer** base class is derived from a more general **Iterator** class.

ApplicationInterface: This abstraction defines the interface between an Optimizer and a simulator program. It encompasses the specific details of a given engineering application. Everything external to this interface is generic and independent of the problem being solved. An ApplicationInterface object will *use* both an IFilter object and an OFilter object to accomplish its task.

IOFilter: A utility class abstraction used by the ApplicationInterface class to provide communication links between the generic data formats used by the Optimizer and the specific input and output formats of a particular simulator program. The input and output operations are logically similar but separable enough to warrant two derived classes (IFilter and OFilter).

OptParameter: A vector of floating point values representing the parameters being optimized.

OptResponse: An abstraction for storing the desired output data set of a simulation. OptResponse contains values for the objective function, constraints, and (in some applications) sensitivities.

Object-oriented techniques such as inheritance and polymorphism are being exploited so that the abstract objects are easy to use and sufficiently generic to encompass a wide variety of problems. Having properly designed the interface, the mapping of parameters to responses shown in Figure 1 provides generic information to the optimizer, and the application and implementation specifics are hidden. The result is a flexible, reusable, and robust multi-disciplinary toolkit that establishes the protocol for wrapping parameter optimization around computationally-intensive finite element analyses.

Optimizer iterators are part of a larger "iterator" hierarchy in the DAKOTA system. In addition to optimization algorithms, the DAKOTA system is designed to accommodate nondeterministic simulation and parameter study iterators. Other classes of iterator methods may be added as they are envisioned, which "leverages" the utility of the Application Interface development.The inheritance hierarchy of these iterators is shown in Figure 2. Inheritance enables direct hierarchical classification of iterators and exploits their commonality by limiting the individual coding which must be done to only those features which make each iterator unique.

Several optimization algorithm libraries and strategies are inherited from the **Optimizer** base class. DOT (Vanderplaats Research and Development 1995), NPSOL (Gill et al. 1986), OPT++ (Meza 1994), and SGOPT (Hart 1994, Hart 1995) have been incorporated in this framework as libraries of *stand-alone* optimizers. Additionally, the "Hybrid" and "SAO" optimization strategies are *combination* strategies which have been conceptualized. In the former, two or more stand-alone optimizers are combined in a hybrid strategy. For example, a coarse-grain genetic algorithm might initially be used to locate promising design space regions, followed by the use of nonlinear programming to converge efficiently on local optima. Effective switching metrics are a key concern. In the latter, a

stand-alone optimizer is interfaced with a separate function approximation toolbox in the setting of sequential approximate optimization (SAO, see (Haftka et al. 1990)). Here, the accuracy and expense of the approximate subproblems, the mechanisms by which the approximations are updated, and the mechanisms of move limit enforcement are key concerns.

Software development work is ongoing. In addition to extension of iterator capabilities and incorporation of additional simulator programs through input and output filter development, general software infrastructure extensions are being implemented in the areas of active set strategies, support of analytic sensitivities, advanced problem specification and system control, and general restart capabilities.

Applications and Results

In application work, the targeted technology areas are nonlinear large-deformation solid mechanics, heat transfer, fluid mechanics, and structural dynamics.

Nonlinear Mechanics: Shape Optimization of a Hazardous Material Transportation Cask

Problem Description. Design of hazardous material transportation casks is an area where numerical optimization can have a large and immediate impact (Harding et al. 1995). These casks are used to transport spent nuclear fuel, high-level waste, and hazardous material. Since they transport such materials, their design and certification must adhere to strict Nuclear Regulatory Commission regulations. A typical transportation cask is shown in Figure 3. There are several components that constitute a typical cask, including impact limiters, containment vessel, shielding, and closure mechanisms, whose designs could be numerically optimized.

In the past, typically, each component was designed separately based on its driving constraint and the expertise of the designer, the components were assembled, and then modified until all of the design criteria were met. This approach neglects the fact that, in addition to its primary function, each component can also have secondary purposes. For example, an impact limiter's primary purpose is to act as an energy absorber and protect the contents of the package, but it can also act as a heat dissipater or an insulator. However, designing the component to maximize its performance with respect to both objectives severely convolutes the problem. Numerically-based optimization schemes can readily attack such problems in an efficient manner. Thus, since the design of these packages involves a complex coupling of structural, thermal and radiation shielding analyses and must follow very strict design constraints, numerical optimization provides the potential for more efficient and robust container designs.

The container weight is to be minimized with respect to shape design variables, subject to design constraints on the cask performance in fire and impact accident simulations. The shape of the cask has been parameterized with respect to 6 design variables (x_1 - x_6) which control the thicknesses of 3 overpack layers in the radial and axial directions (Figure 4). Lower and upper bounds for each of these 6 design variables are 0.1 inch and 20 inches, respectively. The finite element model in Figure 4 shows a simplified geometry over that of Figure 3 in which the stainless steel overpack closure and the locking mechanisms are neglected. The 3 overpack layers consist of 1 layer of aluminum wire mesh impact limiter (density = 448.5 kg/m^3) sandwiched between 2 layers of ceramic cloth thermal insulation (density = 801.0 kg/m^3). The optimizer competes these overpack layers against each other based on relative weight and effectiveness in satisfying the certification constraints. Three separate accident conditions are of concern, each supplying a design constraint on the optimization. In the first accident scenario, a 500 kg steel plate is dropped from a height of 9 m onto the end of the cask which is supported on a rigid foundation (the "end-on" impact; Figure 4). The nonlinear mechanics code PRONTO2D (Taylor and Flanagan 1986) is used to determine stress histories for a given cask design, and for the end-on impact, a design constraint enforces an allowable of 23,000 psi on the maximum axial stress (σ_{yy}) in the inner container. Second, for the same plate impact in a side-on configuration (Figure 5), a design constraint enforces an allowable of 8,440 psi on the maximum inner container axial stress (σ_{yy}). These stress allowables were obtained through calibration to a highly refined model, in which a detailed mesh captured actual threaded seal deformation. Lastly, for a 30 minute 800° C fire, COYOTE II (Gartling and Hogan 1994) is used to generate nodal temperature histories, and a design constraint enforces that the maximum inner seal temperature does not exceed 232° C.

An input filter program was generated to translate the shape design parameters into information used by the PRONTO2D and COYOTE II analysis codes. This requires preprocessing of parameterized template input decks (with APREPRO (Sjaardema 1992)) and automatic mesh generation (with FASTQ (Blacker 1988)). An output filter program computes the weight, reads the stress and temperature time histories (using BLOT (Gilkey and Glick 1989)),

locates the maximum stresses and temperatures, computes constraint values, and returns the objective and constraint function values to the optimizer.

To maximize efficiency with respect to simulation duration while still maintaining the ability to reliably capture the complete impact event, an adaptive termination time strategy was developed and implemented for the impact analyses. This development was motivated by the observation that, whenever event durations vary broadly with design variables, a single selected termination time will invariably be either too long, wasting CPU cycles in continuing the simulation past the event of interest, or too short, terminating the simulation before the peak response is reached and causing inaccurate objective and constraint function evaluations. To avoid the more serious ramifications of the second scenario (underestimation of a critical response), it is common to sacrifice efficiency and adopt a best guess at a sufficiently long simulation duration. Of course, this approach can fail if the variance of event duration over the design space is underestimated; and in fact, an optimizer will naturally seek out those regions of the design space in which the simulation duration is insufficient if the truncation of analyses leads to lower objective functions or more feasible constraint values. A better approach is to adaptively control simulation duration through the monitoring of simulation progress. In impact analyses, event completion can be determined by monitoring the kinetic energy (KE) time history for rebound (an increase in KE following the near-zero minimum state), after which the simulation can be terminated with a Unix kill process command. The KE monitoring and process kill is accomplished with a Unix background process which is launched from the PRONTO analysis driver (see Figure 1) and which cycles with a sleep-delay. This strategy was highly effective in conserving compute time while guaranteeing capture of the peak stress.

Optimization Results. Nonsmoothness of response variations with respect to design variables is troublesome for nonlinear programming techniques, especially when sensitivities are obtained by finite difference. For the end-on vertical impact analysis, continuous improvements in analysis, including model refinement, filtering of stress time histories, increases in platform precision (from single to double precision FORTRAN), and modifications in contact line treatment have been necessary in order to minimize nonsmoothness. Figure 6 shows sequential improvement in design space smoothness for variation in maximum axial stress (σ_{yy}) with respect to fine changes in vertical impact limiter thickness (design variable x_5), from unfiltered low-precision runs (plot point 'o') to filtered low-precision runs (plot point '*') to filtered high-precision runs on a refined model (plot point 'x') to filtered high-precision runs on a refined model with contact lines node-locked at corners (plot point '+'). This last modeling improvement was needed to remove contact indeterminacies at mesh corners, since these indeterminacies were exciting hourglassing instabilities in the PRONTO stress histories. Clearly, the final response variations provide a far more navigable surface than the initial variations. For the thermal analysis, similar refinements have been required, as shown in Figure 7. Mesh refinements, time step size decreases, and the tightening of iterative solver convergence tolerances were required to progress from the initial stair-stepped curve through the sinusoidal curve to the final, relatively smooth, response variation. These nonsmoothness reductions in the impact and thermal analyses were required to allow for effective design space navigation with gradient-based optimizers.

With the bulk of the troublesome nonsmoothness removed, optimization studies have been successful in minimizing the cask weight with respect to the end-on constraint, the thermal constraint, and all three constraints in the combined problem (no meaningful stand-alone weight minimization problem exists for the side-on impact model since x_4, x_5, and x_6 are not defined; see Figure 5). Minimum weights and constraint values are shown in Table 1 (active constraints are underlined), the associated design variable values are shown in Table 2 (variables at or near their bounds are underlined), and the optimal shapes are graphically compared to the geometry of a successful experimental prototype in Figure 8. In Table 1, as would be expected, the thermal and end-on optimum designs are active on their respective constraints. The combined optimum is active on the thermal and end-on constraints, and inactive on the side-on constraint. In Table 2, it can be seen that the end-on constraint primarily drives axial stroke length (x_4, x_5, and x_6) and radial wire mesh thickness (x_2), and the thermal constraint primarily drives radial thickness of the thermal shields (x_1 and x_3). In the combined design, it can further be seen that the outer thermal shield is redundant (x_3 and x_6 go to their 0.1 inch lower bounds). The fact that the necessary thickness of thermal shield belongs in the innermost layer is intuitive, since the thermal shield is heavier than the wire mesh and gains no obvious thermal advantage in being positioned further out radially. The wire mesh, on the other hand, is lighter, pays a smaller weight penalty for being the external layer, and gains a mechanical advantage in being further separated from the centroidal axis. The 94 lb. combined optimum design is a substantial improvement over both the successful experimental prototype containing 150 lbs. of overpack (Figure 8) and the previously published best design of 184

lbs. (Witkowski et al. 1994).

Technical Lessons Learned. The application of optimization to this problem led to several observations:

- Shape optimization with automatic remeshing is a tricky business, particularly when design variables are inclined to seek their lower bounds. Poor element aspect ratios can promote numerical instabilities in nonlinear solvers, which must ultimately be addressed with increased mesh density. That is, as a shape design variable approaches its lower bound, the characteristic element size in the region defined by the shape variable must decrease in order to accurately compute the desired responses. This can dramatically increase the compute time, both directly, through the increase in total degrees of freedom, and indirectly, through the determination of stable time step sizes.
- An effective balance of nonsmoothness vs. CPU has to be determined in many engineering simulations. One must have navigable objective and constraint function surfaces, but must also have tractable CPU usage in the simulations. Clearly, these are competing factors, since refining the modeling controls and mesh density invariably increases the CPU usage of a simulation. Furthermore, the level of required smoothness may be a function of the type of optimizer being used. This fact is further born out in the following heat transfer application.
- Adaptive simulation termination strategies increase both the efficiency and the robustness of optimization studies, by conserving compute time while still guaranteeing capture of the peak responses of interest.

Heat Transfer: Determination of Worst Case Fire Environments

Problem Description. In thermal science simulations, parameter sets are sought which produce worst-case credible fire environment(s) for which structures and systems (such as aircraft, weapons, or petrochemical processing plants) must be designed. These inverse problems can be solved within an optimization framework. As a demonstration, optimization techniques have been applied to determine the vulnerability of a safing device to a "smart fire" (Romero et al. 1995). The optimization parameters consist of the location and diameter of a circular spot fire impinging on the device. The temperature of the fire is constant, though the heat flux it imparts to the device varies in time and space coupled to the response of the device. Function evaluations involved transient simulations using a nonlinear QTRAN (PDA Engineering 1993) heat transfer model with radiative and conductive heating. The finite element model used in the analysis is shown with typical temperature contours in Figure 9. Each simulation required 20 CPU minutes to solve (at tight error tolerance levels, see Figure 13) on a node of an IBM SP2 workstation.

The components of interest must work together to prevent the device from operating except under the intended conditions. It is a weaklink/stronglink design: the weaklink is designed to fail under adverse conditions, which renders a potential stronglink failure incapable of harm. The weaklink is a Mylar-and-foil capacitor winding mounted on the outside of the safing device and the stronglink is a stainless steel plate mounted inside a cavity and offset right of center as shown in Figure 9. The time difference between failure of the stronglink and failure of the weaklink is the safety margin for the device and varies with the fire exposure pattern on the device surface. Hence, to validate the design of the safing system, the worst-case fire exposure pattern is sought by using optimization to minimize this safety margin for selective exposure to a 1000O C black body heat source.

Typical critical node temperature histories are shown in Figure 10 for a 20 hour fire exposure, where the critical node of a link is the one which reaches its failure temperature earliest. The safety margin shown graphically is the objective function that the optimizer minimizes with respect to the design parameters of fire spot-radius (r) and fire center location (x), subject to simple bounds ($0.5 \le r \le 5.8$, $-2.9 \le x \le 2.9$).

An input filter program was generated to translate the optimization parameters into information used by the QTRAN analysis code. Node, element, and surface information are obtained from the PATRAN (PDA Engineering 1988) neutral file description of the finite element model. The heating load applied to each exposed element is then calculated, and this load is assigned to the corresponding nodes of the mesh. QTRAN is executed to determine the temperature histories of monitored nodes for up to a 200 minute exposure to these heating conditions. The QTRAN simulation rarely runs for the full 200 minute duration because of the use of an adaptive termination strategy. This strategy uses a background script to monitor the simulation progress periodically and to execute a kill command once failures have been captured for both links. This procedure is used to reduce the computational expense of the simulations. Once the QTRAN simulation is completed, an output filter program reads the nodal temperature histories and calculates a failure time for each node. In this calculation, interpolation between time step values is used to accurately estimate captured link failures, and if the analysis ran the full 200 minute duration without capturing failures for both links, then a linear extrapolation in time is used to approximate the uncaptured link failures. The

earliest weaklink nodal failure time is then subtracted from the earliest stronglink nodal failure time, and this objective function value is returned to the optimizer.

Optimization Results. This application was challenging from an optimization perspective due to the nonsmoothness and nonconvexity of the design space. In Figures 11 and 12, one-dimensional parameter studies show evidence of multimodality (Figure 12) and of slope-discontinuity at the minimum (Figures 11 and 12). The slope-discontinuity in the figures is caused by switching of the critical weaklink node between geometric extremes. Figure 11 shows negative curvature near the discontinuity (nonconvexity), whereas the discontinuity in Figure 12 is more difficult to discern since the curvature is positive near the discontinuity.

These two parameter studies also provide insight into the mechanics of the problem. In Figure 11, the offset of the lowest safety margin from x=0 (the center of the device) backs up engineering intuition in that the stronglink is also offset right of center. That is, a fire centered roughly over the stronglink causes a lower safety margin. Figure 12 shows a less intuitive result, in which it is evident that the lowest safety margin is not achieved with either a large fire or a small fire. Rather, there exists an insidious, medium sized fire which is not so small that the heating rate is insufficient and which is not so large that it prevents selective heating.

Figure 13 is a detail of Figure 11 and shows evidence of small-scale nonsmoothness, which was reduced through the tightening of QTRAN convergence tolerances at the cost of approximately an order of magnitude greater computational time per analysis. EPSIT and EPSIT2 are absolute convergence tolerances in degrees which govern time step completion and node inclusion in nonlinear iterations, respectively. The additional computational expense per analysis was warranted in this case since none of the nonlinear programming algorithms could successfully navigate the design space without reducing the nonsmoothness (even with large finite difference step sizes).

Several nonlinear programming optimization packages were employed for the solution of this problem, including DOT, NPSOL, and OPT++. In general, the Newton-based optimizers (NPSOL's sequential quadratic programming algorithm, DOT's BFGS quasi-Newton method, and OPT++'s quasi-Newton methods) performed poorly due to inaccuracy and ill-conditioning in the Hessian approximation caused both by the nonconvexity of the design space and by the relatively large finite difference step sizes needed to overcome the small-scale nonsmoothness. Conjugate gradient (CG) methods (DOT's Fletcher-Reeves CG and OPT++'s Polak-Ribiere CG) were much more successful. Furthermore, choice of finite difference step size (FDSS) for computation of gradients proved to be important. Table 3 shows the results for CG optimizers with varying FDSS and with EPSIT=10^{-2} and EPSIT2=10^{-4} (see Figure 13 for effect of EPSIT tolerances). An asterisk (*) in the function evaluations column indicates that the optimization terminated prematurely due to a search direction that failed a descent direction test.

The first four rows of Table 3 illustrate the effect of FDSS on the optimization: FDSS should be as small as possible to allow for effective convergence to a minimum (0.1% is better than 1% which is better than 4% since the gradients are less accurate locally for larger FDSS), but still large enough that small-scale nonsmoothness does not cause erroneous gradients (0.01% is too small; the optimizer cannot successfully navigate the design space since the FDSS is on the order of the design space noise). The last five rows show that DOT's CG optimizer was more robust and more efficient than OPT++'s CG optimizer, through the fact that DOT was successful from 3 different starting points and OPT++ from only one, and through the lower number of function evaluations that were required. The chief cause for these differences was not the version of CG being used (in fact, Polak-Ribiere is generally regarded to be superior to Fletcher-Reeves (Coleman and Li 1990)), but rather was DOT's robust line search routine. OPT++'s line search routine was tuned for efficiency in smooth applications and was overly aggressive in this application; the OPT++ line search library has since been extended to include more robust routines for nonsmooth applications.

To achieve the best answer possible, the QTRAN convergence tolerances were tightened 2 additional orders of magnitude (EPSIT=10^{-4}, EPSIT2=10^{-6}) and the FDSS was reduced to 0.1%. DOT's Fletcher-Reeves CG algorithm was used to obtain the lowest objective function value of 2.5309 minutes (r=1.6204, x=0.78205) which, when compared to stronglink and weaklink failure times of 62.743 and 60.212 minutes respectively, corresponds to a safety margin of just 4%. The 2.5 minute safety margin is an order of magnitude lower than anticipated prior to the study, meaning that the safing device has been shown to be much more vulnerable than was expected. With this relatively low safety margin, it becomes crucial to assess the effects of nondeterminism in the model, which, in the future, can be accomplished with minimal additional effort by "instantiating" an iterator from the **Nondeterministic** base class (see Figure 2). Thus, optimization has successfully solved the difficult problem of worst-case design safety and suggests that design improvements may be warranted.

Pattern search optimizers from the SGOPT package have also been tested on this application. Preliminary results have shown that the same minimum safety margin of 2.5 minutes can be reliably achieved with pattern search.

Furthermore, since pattern search is less sensitive to small-scale nonsmoothness than gradient-based techniques, looser EPSIT tolerances can be employed in the fire simulations which lowers the individual simulation expense considerably. Initial studies have shown that, even though pattern search usually requires more function evaluations than gradient-based optimization, the lower individual simulation expense more than compensates, making the overall computational expense of the pattern search optimization lower than that of the gradient-based optimization (see (Eldred 1996) for a more detailed discussion). Pattern search is, however, susceptible to the "curse of dimensionality," meaning that the method becomes less competitive in efficiency as the number of design variables increases.

Technical Lessons Learned. The application of optimization to this problem led to the following observations:

- When using finite difference gradients in applications with nonsmoothness, an effective finite difference step size is not easily determined. It must be as small as possible to allow for efficient convergence, but still large enough to not be adversely affected by small-scale nonsmoothness.
- Hessian ill-conditioning can be a problem in nonconvex design spaces, causing poor performance in many Newton-based methods. Trust region methods have the potential to overcome this difficulty while maintaining the theoretical strength of second-order optimizers.
- As in the previous application, when computing transient responses for events of unknown duration, increased optimization efficiency and robustness (compute time is conserved, desired response capture is guaranteed) can be achieved with use of an adaptive simulation termination strategy.
- Analysis code convergence tolerances can have a substantial effect on the local nonsmoothness in the design space. When using a gradient-based optimizer, it may be necessary to pay the increased computational expense to employ tight tolerances, so that the optimizer can navigate the design space effectively.
- Gradient-based optimizers put substantial faith in the accuracy of the computed search direction, and in nonsmooth applications, this level of faith may not be justified since the gradients used to calculate the search direction have questionable accuracy. Pattern search optimizers do not confine themselves to a single search direction, but rather search multiple directions simultaneously. As a result, they can be more robust in nonsmooth applications. Moreover, efficiency can be comparable when the number of design variables is small.

Fluid Mechanics: Coating Flow Die Design

Problem Description. The manufacture of polymeric thin-film coatings is an expensive process, requiring high-maintenance, close-tolerance tooling which must be frequently replaced. In premetered processes such as slide, slot, and curtain coating flows (the primary manufacturing methods for film products like video tapes and color photographic film), liquids are distributed uniformly in the transverse direction by chamber-slot dies. Dies must be carefully designed and machined with tight tolerances to ensure uniform flow at the exit (transverse nonuniformity must fall within a few percent). The die-fabrication process is costly in both dollars and time, but is necessary to maximize production throughput of high-quality thin films at the lowest cost. A typical thin-film die is illustrated in Figure 14. Polymer is pumped into a chamber through a feed pipe, and is then extruded through a slot. Typical slot dimensions are 5' wide by .01" high. The output flow profile is typically laminar and the fluid is assumed to be Newtonian. The flow is highly sensitive to a number of design parameters, including geometrical parameters (e.g., slot dimensions, chamber shape and size), fluid properties (e.g., liquid viscosity and density), and process conditions (e.g., volumetric flowrate).

The goal of this study is to find the optimum combination of die geometry parameters which minimizes nonuniformities in the output flow profile for a given set of fluid properties and operating conditions. The objective function of outflow plane nonuniformity is computed by integrating the velocity profile over the outflow plane and is formulated as the percent of coating material across the slot width whose deviation is greater than 1% from the mean velocity. "Perfect" uniformity (0% nonuniformity) is not achievable as the problem is formulated due to the "no-slip" condition at the flow boundaries (i.e. walls of the die). That is, some portion of the coating width will always exceed 1% deviation due to the fact that the coating velocity profile must ramp up from zero at the wall. The die design geometry shown in Figure 14 is parameterized using six design variables that vary the pre-determined sensitive dimensions of the die. These dimensions are slot length L_s, slot height H_s, slot entrance angle α, chamber length L_c, chamber height H_c, and feed-channel height H_f. Slot width W is fixed at 12.7 cm. Constraints on the problem include pump pressure range, an upper bound on the average residence time, and a tolerance band on process temperature. Higher pump pressures generally cause larger flowrate or velocity fluctuations in the feed channel, and higher power (i.e. $) requirements. More importantly, excessive pump pressure can cause deflection of the slot walls, which in turn

results in variation of the slot gap height and thus an undesirable increase in flow nonuniformity at the slot exit. In temperature-sensitive coating flow applications (e.g. hot melt extrusion), process temperature must be maintained within a tolerance band to avoid changes to the fluid properties and to the mechanical stability of the die. In low viscosity applications, temperature is less important since operations are normally carried out at room temperature.

There are two stages in a premetered coating flow. The first stage involves pumping coating liquid into a die distributor (or distributors when simultaneous multilayer coating flows are carried out) and distributing the coating liquid uniformly across the die width. In this stage, flow boundaries are of fixed type, i.e. no free boundaries are present. Accordingly, flow simulation is straightforward. The second stage involves transferring the coating liquid from the coating head (i.e. die) to the fast-moving flexible substrate (or web). In this stage, not only are free surfaces (i.e. air/liquid interfaces) present, but the flow also contains static and dynamic contact lines. Here, flow simulation is much more challenging. At present, no commercial code can satisfactorily simulate this latter stage of premetered coating flows. To fill this void, researchers at Sandia National Laboratories are developing specialized computer codes that can efficiently simulate the second stage of a variety of coating flows. For the present study, which involved three-dimensional flow with fixed boundaries, a commercial code based on the finite element method, namely FIDAP (Fluid Dynamics International 1993), was employed.

An input filter program was generated to translate the die geometry parameters into information used by the FIDAP analysis code. This requires preprocessing of parameterized template input decks with APREPRO. Following the FIDAP analysis, an output filter program reads the velocity and pressure profiles from the FIDAP output, integrates the velocity data in calculating the objective function, computes the constraint values based on residence time and maximum pressure calculations, and returns the objective function and constraint values to the optimizer.

Optimization Results and Technical Lessons Learned. As discussed in the earlier applications, nonsmoothness of response surfaces is critical not only for a gradient-based optimization scheme to be efficient, but for it to be viable at all. In this application, two different finite element models were used to investigate the response surface (nonuniformity in the outflow plane versus geometric parameters) smoothness issue. The first model (the "coarse model") contained 540 elements, whereas, the second model (the "fine model") had 980 elements. Both models provided navigable surfaces; however, the coarse model had kinks in its surfaces which were nonexistent using the fine model. Although the fine model had more than twice the computational burden, it was necessary to pay the increased computational expense to insure a smoother surface for the optimizer to navigate. The initial geometry and finite element mesh for the fine model are shown in Figure 15.

In the case study presented here, low viscosity coating liquids typically used in precision premetered coating processes are of interest. Specifically, fluid viscosity and density were chosen to be 15 cP and 1000 kg/m^3 respectively, volumetric flowrate per unit slot width was set to be 1.5×10^{-4} m^2/s, and a characteristic length scale was chosen to be 10^{-3} m. With these conditions, the resultant Reynolds number is 10. Constraint allowables were set at 2100 for nondimensional maximum pressure (dimensional gauge pressure = 0.7 psi) and 325 for nondimensional average residence time (dimensional time = 2.2 sec). These values were chosen somewhat arbitrarily based on the nominal design to prevent overly large design changes. More realistic process allowables are needed and will be obtained from industry.

DOT's modified method of feasible directions optimizer was used to successfully optimize the die geometry and reduce the nonuniformity from 16.5% to 3.2%. The final geometry is shown in Figure 16. Initial and final die dimensions and constraint values are shown in Table 4 (those dimensions which were driven to their upper or lower bounds are indicated with a '*'). Comparison between initial and final geometries shows that the optimized geometry is about twice as large as at the initial. As expected, the slot height was reduced to its lower bound, thereby causing the maximum pressure to increase to 1832 (dimensional pressure = 0.6 psi). Since neither of the constraints on maximum pressure or average residence time were active, the optimum design computed is not a direct function of the allowables chosen for these parameters. This low pressure optimal solution is typical of low-viscosity coatings. It is expected that the maximum pressures will be considerably greater for high-viscosity coating applications (e.g., adhesive coatings, hot-melt extrusion).

Although numerical optimization has proved to be valuable in improving die design for low viscosity coating flows, it is evident that this is only part of a general die design effort. Continuation of this work will involve investigation of additional coating materials, primarily high viscosity coatings, which will likely require more careful treatment since the operating constraints will be more important in the design. Also, instead of parameterizing with only a slot entrance angle, a more complex parameterization of the die chamber will be used to describe a tear-drop chamber shape. Lastly, efforts are underway to incorporate analytical sensitivity capability within the FIDAP analysis code so that optimization efficiency can be comparatively evaluated for analytical gradient-based, finite difference

gradient-based, and pattern search approaches.

Structural Dynamics: Discrete Optimization of Vibration Isolation Platform

Problem Description. Vibration isolation systems are widely used to protect sensitive devices from vibrations produced in their environment. Typical examples include isolating delicate laboratory experiments from floor-borne vibrations, preventing transmission of vibrations in satellite structures from rotating machinery (e.g., cryo-coolers) to communication antennas or scientific detectors, or isolating a car body or airplane frame from engine vibrations.

Isolation is achieved with the use of passive or active compliant connections between the source of the vibrations and the device to be protected. The classical approach to designing isolation systems focuses primarily on the properties of those compliant mounts without much regard to the geometry of the complete system, that is, the locations and/or orientations of the mounts on the isolated device. In cases where the source of the vibrations is well known (location, amplitude, frequency content) and/or when the residual motion at *specific* points on the isolated device are of critical importance, we should expect (see, for example, (Ashrafiuon 1993)) that mount locations and orientations will have a substantial effect on the effectiveness of the isolation.

To investigate this, we define a simplified vibration isolation problem, based on an existing laboratory setup (Figure 17). The setup consists of a rigid, rectangular optical table (approximately 48" x 36" x 8.5", 815 lb.) mounted on 3 vibration isolation mounts. This system is in turn resting on a massive seismic base (approximately 70" x 48" x 12", 4085 lb.), itself isolated from floor borne vibrations by 4 air bags. The isolator mounts supporting the optical table are steel coil springs with stiffness of about 3970 lb/in in the axial direction and 1570 lb/in in the transverse (shear) directions. The bottom of the optical table and the top of the seismic base feature identical 8x6 arrays of mounting holes for those springs. The 6 rigid body natural frequencies of the seismic base on its air bags range from about 1.1 Hz to 2.5 Hz. Both the base and the optical table can be regarded as rigid bodies below a few hundred Hertz, where flexible modes appear.

A 3-dimensional, rigid body dynamics code was developed in MATLAB (The Mathworks 1992) to model this system. The air bags and steel springs are modeled as 3-dimensional springs with viscous damping. Their bending and torsional stiffnesses are neglected. The seismic base and optical table are modeled as concentrated masses and inertias. The mass properties of all parts of the system were either measured or evaluated from CAD models. The stiffness and damping of the air bags and steel springs were obtained from the manufacturers and further refined through parameter estimation based on measured natural frequencies and modal damping.

To simulate a perturbation, the seismic base is excited at its front right corner by an electromagnetic shaker (Fig. 17) with a sinusoidal input at a fixed frequency of 50 Hz. We assume that the vertical component of velocity at the front left corner of the optical table (Fig. 17) is the critical design criterion (because a sensitive instrument is mounted there for example). These locations introduce asymmetry in the problem and lead to the nonintuitive optimal solutions for the isolator locations.

The design problem consists of selecting locations for the 3 spring isolators on the 6x8 grid of mounting holes in a way that minimizes transmission of the perturbation to the specified point on the optical table. The 6 discrete design variables are the x and y locations of the 3 springs. The transmission T (μin/sec.-lb.) is expressed as the vertical velocity at the given point of the optical table per unit load amplitude at the excitation point. We define a baseline design where the 3 springs are arranged as symmetrically as possible around the center of the table (Fig 18) which has a predicted transmission of 21.22 μin/sec.-lb. at 50 Hz.

Since the optimized designs will be tested in the laboratory, a number of practical constraints apply:

- Due to the presence of lifters, the 4 holes at the corners of the grid cannot be used.
- The 3 springs cannot be colinear and only one spring is allowed per grid location.
- To avoid generating designs that would be too unstable, the first natural frequency of the optical table on its mounts is constrained to be more than 4 Hz. This also ensures decoupling between the modes of the seismic base/airbags system and those of the optical table/springs system.
- Because the springs are simply resting in end plates attached to the mounting holes, the static load on each spring must be compressive.
- There is a limit to the amount of weight a spring can support before being compressed to its solid length. An upper limit is imposed on the static deflection of each spring.

Optimization Results. Because of the discreteness of the problem, the optimization was performed with the genetic algorithm (GA) available in SGOPT within DAKOTA. The MATLAB analysis code was coupled to the optimizer through simple UNIX scripts. No input or output filters were used; instead, the MATLAB code was designed to

exchange input and output directly in the DAKOTA-compatible format. The design variables (locations of 3 springs) are coded into a "chromosome" composed of 6 integer-valued "genes" that define the grid indices for x and y coordinates of the first, second and third spring, i.e. [x1, y1, x2, y2, x3, y3]. Any x gene can take integer values between 1 and 6 while y genes take values between 1 and 8. The size of the design space (total number of conceivable designs) is approximately 100,000.

Since the GA in DAKOTA cannot explicitly handle constraints, penalty functions are implemented within the MATLAB analysis code. Both step and quadratic penalty functions were used for this problem. In the step penalty, any amount of constraint violation produces a fixed amount of penalty, subtracted from the fitness (Leriche 1994). This is done to ensure convergence to a feasible design, which may not occur when using only progressive penalty functions. Because GA's are zero-order search techniques, the discontinuities this strategy creates are perfectly acceptable. In addition to the step penalties, the "quantifiable" constraint violations (4Hz frequency limit, spring static loads and deflections) produce penalties proportional to the square of the violation (constraints are first normalized).

Out of several options and adjustments available in DAKOTA, the following combination was chosen:

- population size: 10; crossover probability: 0.8; mutation probability: 0.1; number of generations: 15.
- ranking technique for selection: the probability of selection of an individual is proportional to its rank.
- moderate selection pressure: the probability of selection of the best individual is twice that of the worst.
- uniform mutation: a mutated gene takes a random integer value, uniformly distributed from 1 to 6 for x genes and 1 and 8 for y genes.
- elitist strategy: the best individual of the current generation is duplicated into the next generation.

With these adjustments, each search evaluates less than 150 designs, or 1.5% of the design space (the GA keeps track of previously analysed designs, so the actual number of function evaluations averages around 105). In order to get some idea of the reliability of the search, we performed series of 10 runs and compare the results to a series of 10 random searches of 105 analyses each. The result from each random search is the best feasible design found. Typical results are shown in Fig. 19. The random searches generate some good designs and many mediocre ones. In contrast, all 10 designs obtained from the GA runs are feasible and represent significant improvements from the baseline case. However, the GA occasionally "converges" to a relatively poor design (T=3.23 in Fig. 19). This shows that more reliable results can be obtained by running a small number of short searches (if the probability that the best found design is "poor" is 0.1 for a one run, then it is only 0.01 for the best of 2 runs, 0.001 for the best of 3, etc.). Because GA's are most efficient in the initial phases of the search and further "convergence" is usually slow, this approach is preferable to running a single longer search (Leriche 1994).

The "optimal" designs are also very different from each other, as illustrated in Fig. 18 where a few configurations are shown with their predicted transmissibility. This is a strength of genetic algorithms: final designs are random "quasi-optimal" configurations that tend to have similar values of the objective function but distinct design features; the final choice rests in the hands of the designer. However, in this case, the various designs of Fig. 18 resulted from distinct runs of the GA and were not present together in final populations. In fact, the final populations usually contained only one or two "good" designs, with most others infeasible. This is the result of 1.) significant multimodality, as evidenced by the large differences between distinct "optimal" designs (Fig. 18), 2.) a highly constrained design space (12 constraints), and 3.) a relatively small design space (100,000 designs).

This creates small "pockets" of feasible designs isolated from each other in the design space. Because the design space is small (coarse 6x8 grid) and feasible designs are rare (Monte Carlo simulations show that only about 13% of random designs are feasible), each "pocket" tends to contain only a few designs and the probability that genetic operators will generate feasible designs is small (even by mating two feasible designs). Also, since designs in distinct pockets are very different, there is no reason to expect that combining features of those different designs would create better or even feasible designs (the concept of niches (Goldberg, 1989) was developed to handle this problem but we think that in this case, the number of feasible designs in each niche is too small to allow exploitation by GA operators). This means that most of the search has to take place in the infeasible design space and explains the presence of few feasible designs in each population.

Continuous optimization was also tested on this problem. DOT's modified method of feasible directions was used within the DAKOTA framework to solve the constrained continuous problem (ignoring the grid), followed by rounding of the optimal design to neighboring discrete solutions. An optimal solution was found at (2.22, 1.56, 2.49, 4.94, 4.29, 3.79) with a transmission T=0.20 μin/sec.-lb. Rounding to the closest neighbor gives (2,2,2,5,4,4), which is infeasible. If we consider all immediate neighbors of the continuous solution (Fig. 20), we find that out of the 64 designs, only 12 are feasible and the best of these has T=3.67, 22 times worse than design A of Figure 18.

Figure 21 shows frequency response functions (FRF's) for the designs of Fig. 18. These FRF's show that the GA is seeking out an anti-resonance condition in the vicinity of 50 Hz. Anti-resonance is achieved by combining mode shapes so as to cancel out vibration at a point. The fact that the anti-resonant peaks miss the 50Hz target slightly is due to the discrete isolator locations; and in fact, the continuous solution of T=0.20 μin/sec.-lb. places the anti-resonant peak directly at 50Hz (not shown). Another important fact to observe is that there is significant broad band improvement in the transmissibilities of the optimized designs compared to the baseline design. That is, the improvements are not confined only to a 50 Hz input; but rather there are significant decreases in transmission through a broad input range. This is an especially important observation, since it shows that the performance is not seriously degraded for off-nominal excitation inputs and that more sophisticated objective function formulations for minimizing broad band transmission are probably unnecessary.

Finally, the designs of Fig. 18 were tested in the laboratory. The results of those tests are shown in Fig. 22 and compared to analytical predictions. The figure shows the nominal value of the transmissibility (objective function in the optimizations) and the predicted range of transmissibilities with 5% uncertainties in the spring constants. This analytical range is determined with Monte Carlo simulations in which each spring constant is chosen randomly within the 5% uncertainty range. Note that, in most cases, the experimental value falls within the analytical range. The transmissibilities of the optimized designs were predicted between 32 and 70 times smaller than that of the baseline design. The actual ratios achieved in the lab range from 7 to 46. They all represent significant improvements from the original design.

Technical Lessons Learned. The application of genetic algorithms to this discrete, multi-modal, heavily constrained problem led to the following observations:

- Applying constraints through penalty functions in a GA problem is a delicate operation. A balance must be achieved between the desire to obtain a feasible final design and the need to allow the search to cross infeasible regions of the design space. Surprisingly little research has been devoted to this aspect. One reason is that, in research GA's, problem-specific repair operators are often introduced to enforce constraints. This approach is more efficient but is highly application-specific and cannot be included in general purpose codes like DAKOTA.
- The combination of multi-modality, large number of constraints, and limited design options (coarse discrete grid in this case) makes the problem difficult to handle, even for a zero-order random search technique like the GA.
- The classical argument that a GA provides multiple design alternatives in its final population does not hold in heavily constrained discrete problems with small design spaces. Instead, each run provides only one or two acceptable designs.
- Multiple design options and improved reliability of the search can be obtained by running a few short searches, rather than a single, long search.
- Continuous optimization followed by rounding to neighboring discrete solutions does not generally lead to an optimal design. For problems with coarse discrete grids, heavily constrained design space, and rapidly varying objective function, this approach leads to few, relatively poor feasible designs.

Conclusions

Object-oriented software design has been shown to be an effective tool for the generic integration of optimization techniques with broad classes of simulation codes. Three applications involved optimizing expensive, nonsmooth engineering simulations with nonlinear programming, and one application demonstrated the solution of a discrete design problem with constraints via genetic algorithm. In several of these applications, the DAKOTA system was used for easy comparison of different optimization algorithms which enabled illuminating assessments of relative performance.

In the engineering simulation applications, nonsmoothness in the design space has been shown to be a recurring problem when applying gradient-based optimization techniques to "black box" transient simulation codes. This ultimately must be addressed with attention to nonsmoothness reductions in the analyses and with employment of robust algorithms in the optimization. The high computational expense of repeated analyses is the other crucial, recurring difficulty. This must be addressed with attention to modeling simplifications, analysis code convergence tolerances, efficient time stop strategies for transient simulations, function approximation strategies, and robust and efficient optimization algorithms which make the most of each function evaluation and which are successful the first time (thereby avoiding multiple executions and tweaking of parameters before achieving convergence).

In the vibration isolation application, the challenges were quite different. Instead of nonsmoothness and extreme

computational expense, discrete design variables and a highly constrained design space were the primary challenges. Genetic algorithms have shown promise as effective techniques for these types of problems. However, while GA's can be very robust techniques, the number of function evaluations they require can be prohibitive for applications involving expensive engineering simulations.

Thus, research in this paper has focused on how generic interfacing of iteration with simulation is performed, what application-specific techniques are useful in enabling reliable and efficient optimization studies, and what algorithmic strengths and weaknesses can be observed when interfacing existing optimization techniques with complex engineering simulations. It has been shown that existing techniques have limitations in their effectiveness when dealing with the combined challenges of high computational expense and nonsmooth, nonconvex design spaces. These observations have uncovered research needs which are directing advanced strategy development efforts in fundamental algorithms, algorithm hybridization, function approximation, parallel processing, and automatic differentiation. Progress in these areas is presented in a separate paper. The overall goal of this collection of research activities is to develop a broadly useful optimization capability with the flexibility and extensibility to easily accommodate broad classes of optimizers, a wide disciplinary range of simulation capabilities, and advanced strategies which seek to enhance robustness and efficiency beyond that which is currently available.

Acknowledgments

The authors gratefully acknowledge the efforts of Dave Harding, Mike Neilsen, Terry Hinnerichs, and Roy Hogan in providing the groundwork for the transportation cask optimization study; Bill Hart in performing preliminary investigations of pattern search optimizers on the worst-case fire surety application; and Clay Fulcher in providing support and technical direction in the vibration isolation application.

References

Ashrafiuon, H., "Design Optimization of Aircraft Engine-Mount Systems," *Journal of Vibration and Acoustics*, ASME, Vol. 115, October 1993, pp. 463-467.

Blacker, T.D., *FASTQ Users Manual Version 1.2*, Sandia Report SAND88-1326, June 1988.

Coleman, T.F. and Li, Y., Eds., *Large-Scale Numerical Optimization*, Society for Industrial and Applied Mathematics, Philadelphia, 1990.

Eldred, M.S., Hart, W.E., Bohnhoff, W.J., Romero, V.J., Hutchinson, S.A., and Salinger, A.G., "Utilizing Object-Oriented Design to Build Advanced Optimization Strategies with Generic Implementation," submitted for the *6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, Sept. 4-6, 1996.

Fluid Dynamics International, Inc., *FIDAP Users Manual*, version 7.0, Evanston, IL, April 1993.

Gartling, D.K., and Hogan, R.E., *COYOTE II - A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part II - User's Manual*, Sandia Report SAND94-1179, October 1994.

Gilkey, A.P., and Glick, J.H., *BLOT - A Mesh and Curve Plot Program for the Output of a Finite Element Analysis*, Sandia Report SAND88-1432, June 1989.

Gill, P.E., Murray, W., Saunders, M.A., and Wright, M.H., *User's Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming*, System Optimization Laboratory, TR SOL-86-2, Stanford University, Stanford, CA, Jan. 1986.

Goldberg, D.E., *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley Publishing Co., Reading, MA, 1989.

Haftka, R.T., Gurdal, Z., and Kamat, M.P., *Elements of Structural Optimization*, Second revised edition, Kluwer Academic Publishers, Boston, 1990.

Harding, D.C., Eldred, M.S., and Witkowski, W.R., "Integration of Finite Element Analysis and Numerical Optimization Techniques for RAM Transport Package Design," proceedings of the *11th International Conference on the Packaging and Transportation of Radioactive Materials* (PATRAM '95), Las Vegas, NV, Dec. 3-8, 1995.

Hart, W.E., *Adaptive Global Optimization with Local Search*, Ph.D. dissertation, University of California at San Diego, May 1994.

Hart, W.E., *Evolutionary Pattern Search Algorithms*, Sandia Report SAND95-2293, Sept. 1995.

Kamat, M.P., Ed., *Structural Optimization: Status and Promise*, Progress in Astronautics and Aeronautics, Vol. 150,

American Institute of Aeronautics and Astronautics, Washington DC, 1993.

Leriche, R., *Composite Structures Optimization by Genetic Algorithms*, Ph.D. dissertation, Virginia Tech, Blacksburg, VA, October 1994.

Meza, J.C., *OPT++: An Object-Oriented Class Library for Nonlinear Optimization*, Sandia Report SAND94-8225, Sandia National Laboratories, Livermore, CA, March 1994.

PDA Engineering, *PATRAN Plus version 2.5 User Manual*, Costa Mesa, CA, July 1988.

PDA Engineering, *P/THERMAL User Manual*, Release 2.6, Costa Mesa, CA, March 1993.

Romero, V.J., Eldred, M.S., Bohnhoff, W.J., and Outka, D.E., "Application of Optimization to the Inverse Problem of Finding the Worst-Case Heating Configuration in a Fire," proceedings of the *9th International Conference on Numerical Methods in Thermal Problems*, Atlanta, GA, July 17-21, 1995.

Sjaardema, G.D., *APREPRO: An Algebraic Preprocessor for Parameterizing Finite Element Analyses*, Sandia Report SAND92-2291, Dec. 1992.

Stroustrup, B., *The C++ Programming Language*, 2nd ed., Addison-Wesley, New York, 1991.

Taylor, L.M., and Flanagan, D.P., *PRONTO2D: A Two-Dimensional Solid Mechanics Program*, Sandia Report SAND86-0594, 1986.

The MathWorks, Inc., *MATLAB User's Guide*, Natick, MA, August 1992.

Törn, A., and Zilinskas, A., *Global Optimization*, Lecture Notes in Computer Science, Springer-Verlag, 1989.

Vanderplaats Research and Development, Inc., *DOT Users Manual*, Version 4.20, Colorado Springs, CO, 1995.

Witkowski, W.R., Eldred, M.S., and Harding, D.C., "Integration of Numerical Analysis Tools for Automated Numerical Optimization of a Transportation Package Design," *5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization* (MDO), paper AIAA94-4259, Panama City Beach, FL, Sept. 7-9, 1994.

Table 1: Weight minimizations subject to end impact, side impact, and thermal constraints.

	Initial Design				Final Design			
	Weight (lbs.)	End σ_{yy} (psi)	Side σ_{yy} (psi)	Max. Temp. (^oC)	Weight (lbs.)	End σ_{yy} (psi)	Side σ_{yy} (psi)	Max. Temp. (^oC)
End-on	181.6	19012.	N/A	N/A	40.40	22976.	N/A	N/A
Thermal	153.9	N/A	N/A	153.4	62.19	N/A	N/A	232.6
Combined	181.6	19012.	4005.	162.6	93.98	23012.	6614.	231.7

Table 2: Design variable values for computed optima.

	x1	x2	x3	x4	x5	x6	Optimal Shape
End-on	.466	.710	.104	1.13	10.5	1.13	Long & Thin
Thermal	1.68	.100	1.04	1.15	.805	.921	Short & Fat
Combined	2.22	.909	.100	.647	10.3	.107	Compromise

Table 3: Optimization results with CG optimizers for EPSIT = 10^{-2}, EPSIT2 = 10^{-4}

Optimizer	FDSS	Initial Values (r, x)	Initial Obj. Fn.	Funct. Evals.	Final Values (r, x)	Final Obj. Fn.
OPT++ P.-R. CG	4%	(1.4, 0.5)	7.2045	34*	(1.5812, 0.75016)	2.8293
OPT++ P.-R. CG	1%	(1.4, 0.5)	7.2045	100*	(1.6038, 0.76547)	2.5956
OPT++ P.-R. CG	0.1%	(1.4, 0.5)	7.2045	73	(1.6086, 0.76895)	2.5546
OPT++ P.-R. CG	0.01%	(1.4, 0.5)	7.2045	28*	(1.6016, 0.57370)	4.9477
OPT++ P.-R. CG	1%	(1.0, 1.0)	86.954	31*	(1.9321, 0.62026)	5.9543
OPT++ P.-R. CG	1%	(1.2, 0.9)	34.831	106	(2.1044, 0.50665)	6.9526
DOT F.-R. CG	1%	(1.4, 0.5)	7.2045	34	(1.6435, 0.77498)	2.5973
DOT F.-R. CG	1%	(1.0, 1.0)	86.954	40	(1.6374, 0.77591)	2.5362
DOT F.-R. CG	1%	(1.2, 0.9)	34.831	38	(1.6475, 0.77393)	2.6217

Table 4: Preliminary Optimization Results

	Initial	Final
Slot Length, L_s (cm)	5.0	11.3
Slot Height, H_s (cm)	1.0	0.25*
Slot Entrance Angle, α (degrees)	53.0	60.0*
Chamber Length, L_c (cm)	10.9	20.0*
Chamber Height, H_c (cm)	4.3	9.9
Feed-Channel Height, H_f (cm)	15.0	14.3
Non-uniformity (%)	16.5	3.2
Average Residence Time†	253.6	215.2
Maximum Pressure†	42.3	1832.

* indicates active bound constraint
† indicates nondimensional quantity

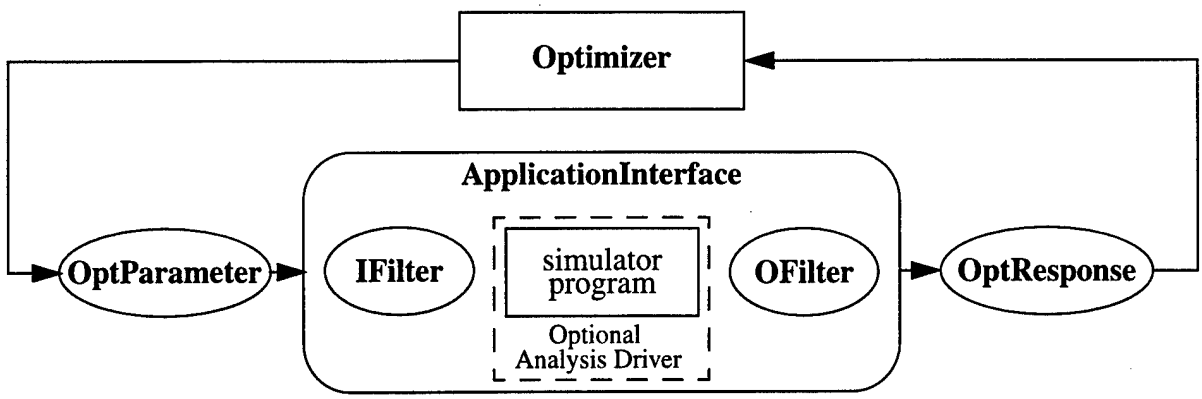


Figure 1. Application interface conceptualization.

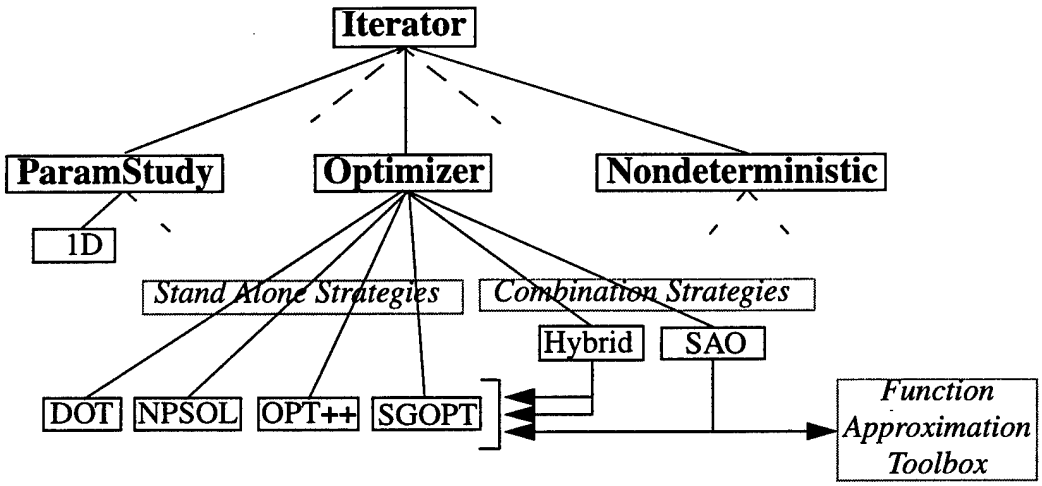


Figure 2. Iterator hierarchy showing broad iteration capabilities:
A) **ParamStudy**: for mapping response variations with respect to model parameters.
B) **Optimizer**: for numerical optimization studies.
C) **Nondeterministic**: for assessing the effect of modeling uncertainties on responses.

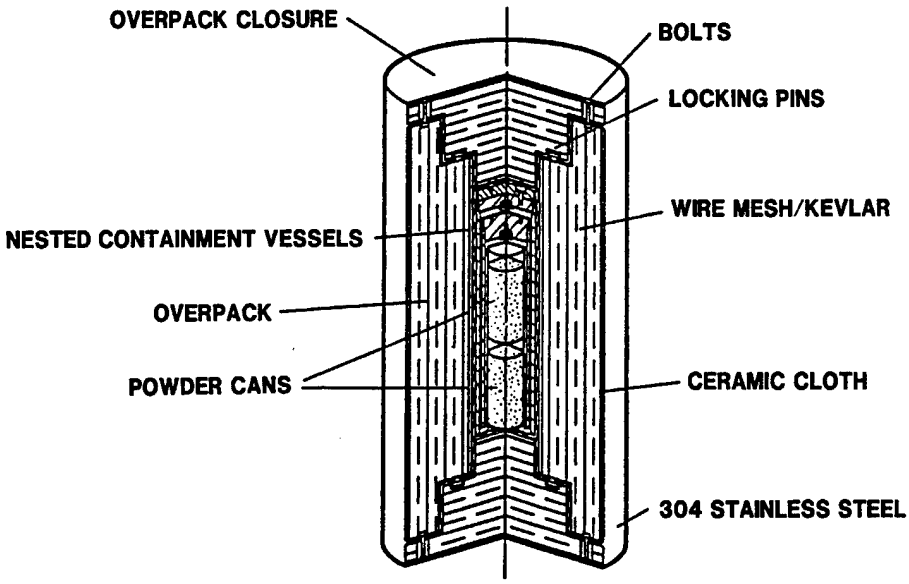


Figure 3. Typical transportation cask.

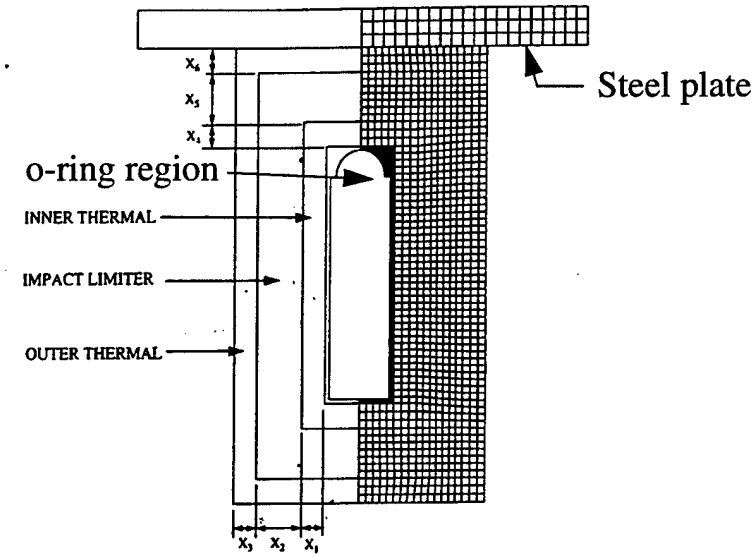


Figure 4. Axisymmetric finite element model for end-on impact showing the 6 design variables which define overpack layer thicknesses.

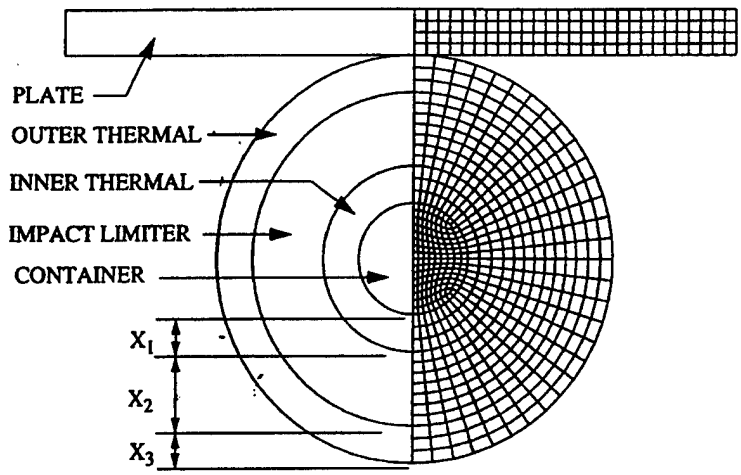


Figure 5. Finite element model for side-on impact showing radial thickness layers.

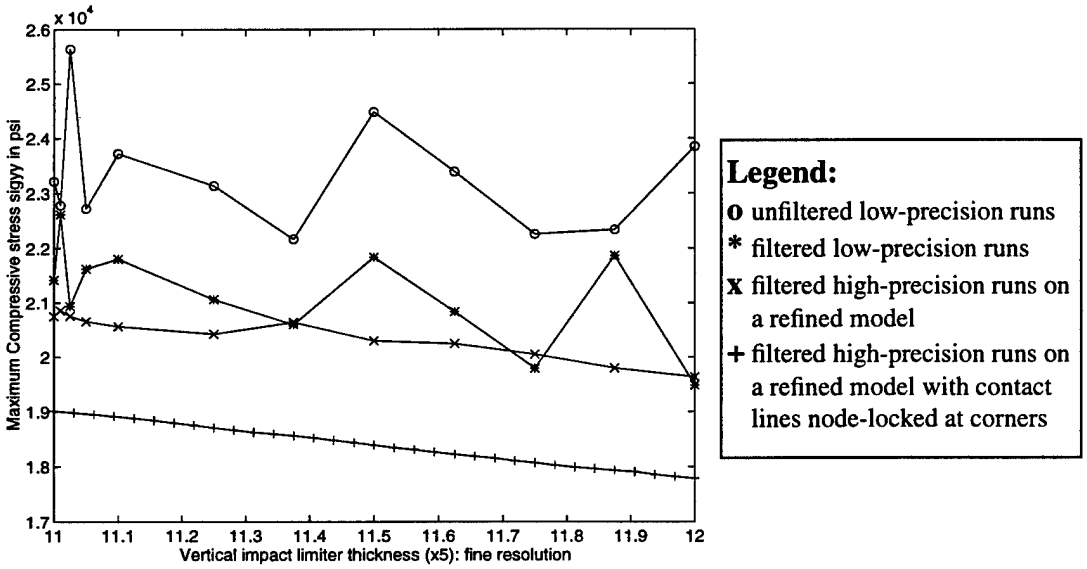


Figure 6. Smoothing of stress constraint through modeling improvements.

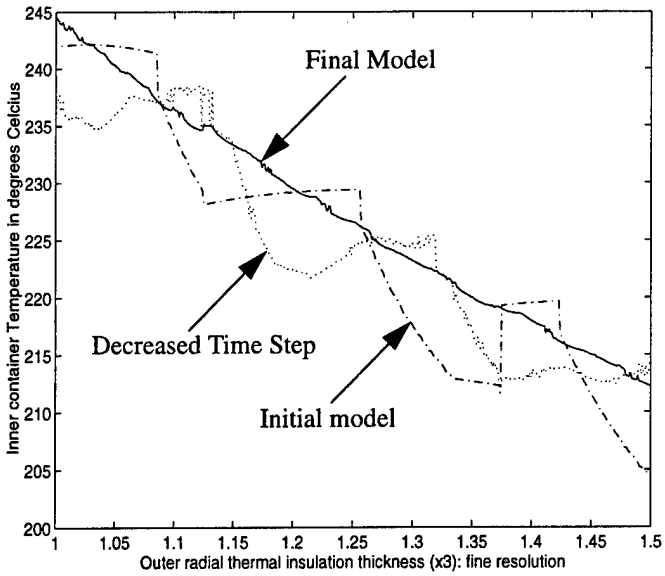


Figure 7. Smoothing of temperature constraint through modeling improvements.

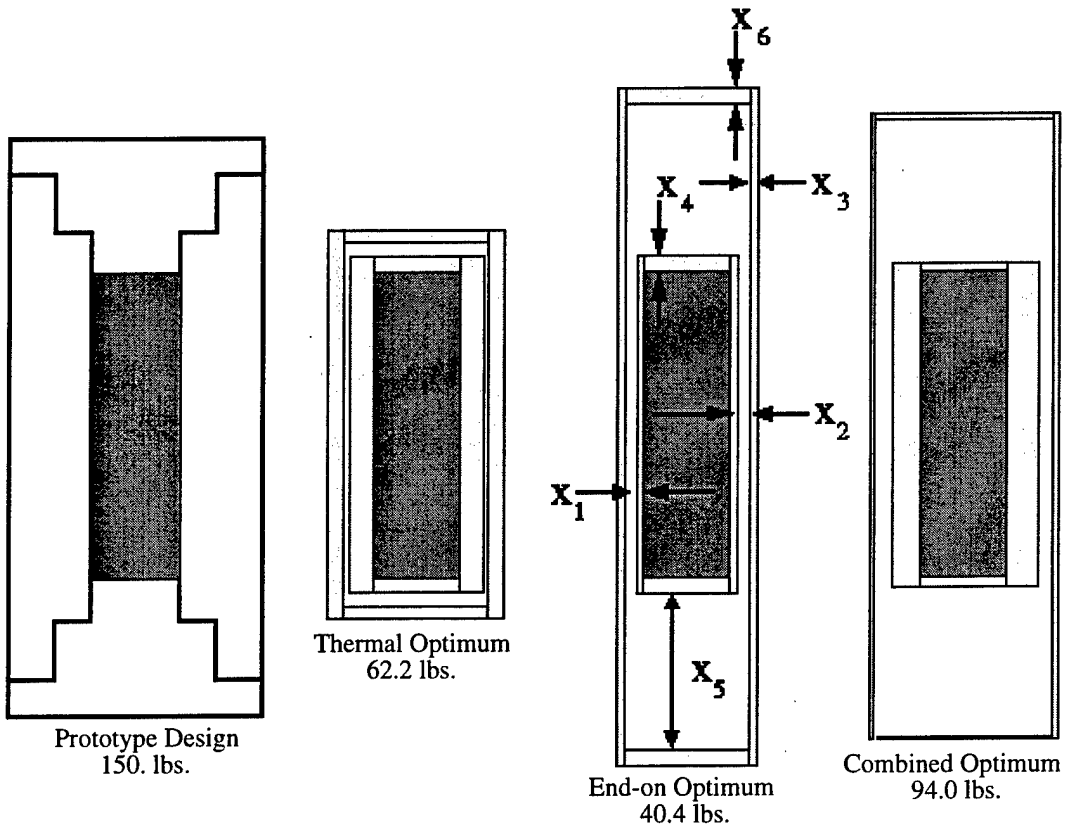


Figure 8. Comparison of computed optimal shapes with experimental prototype design.

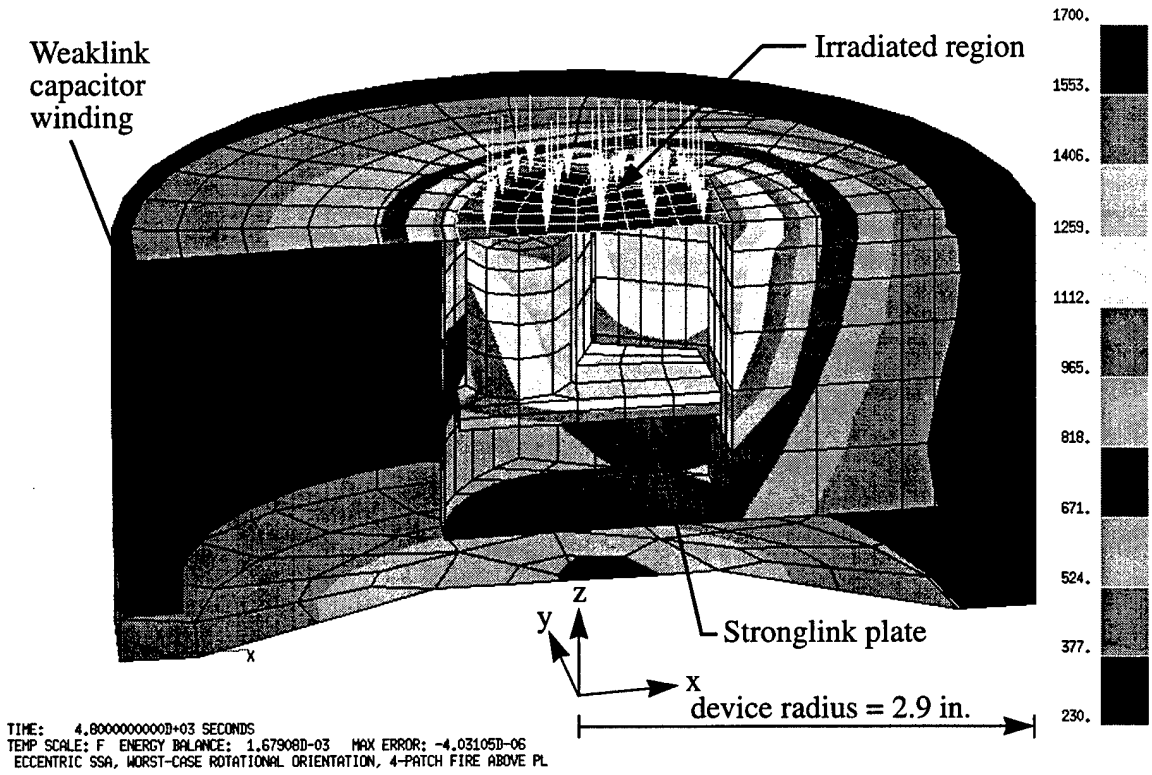


Figure 9. Finite element model and typical temperature distribution (^{o}F).

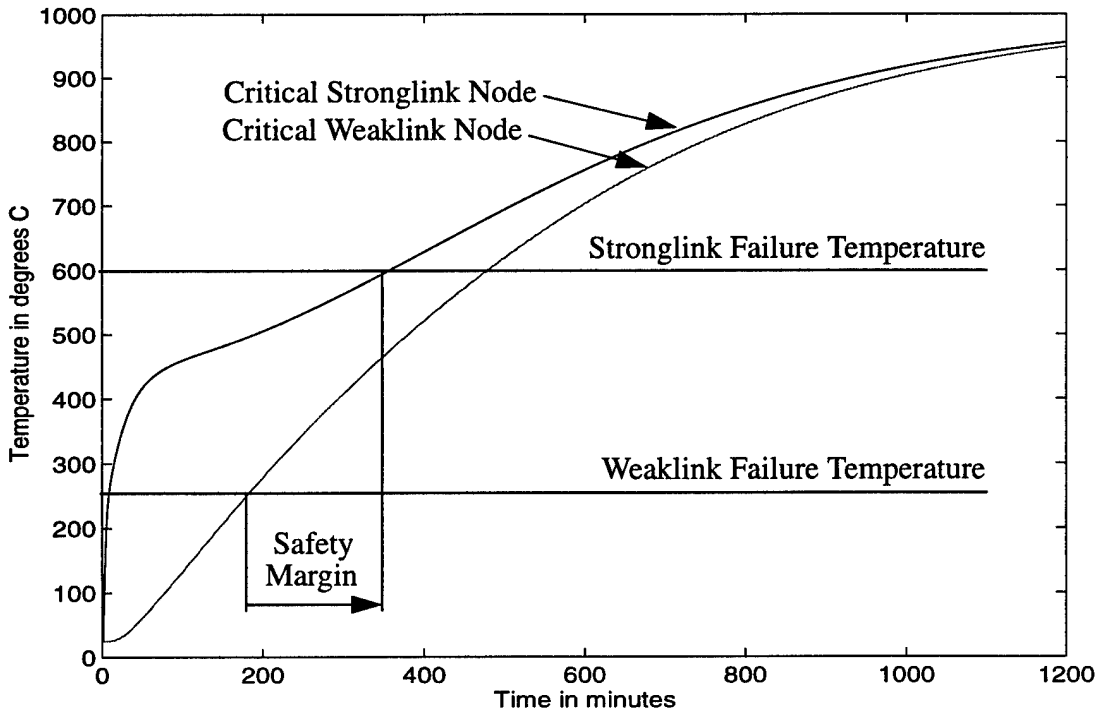


Figure 10. Temperature histories of critical stronglink and weaklink nodes.

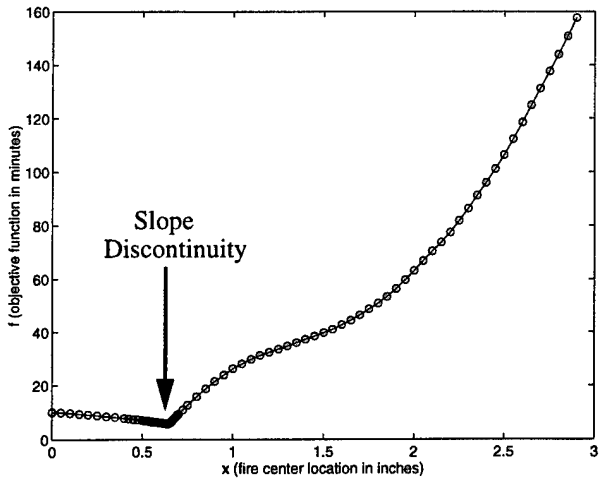


Figure 11. Objective function variation with respect to fire center location (x) for r=1.89.

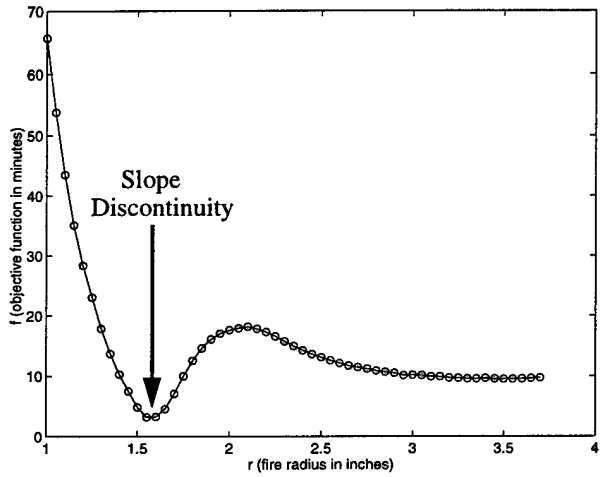


Figure 12. Objective function variation with respect to fire radius (r) for x=0.8.

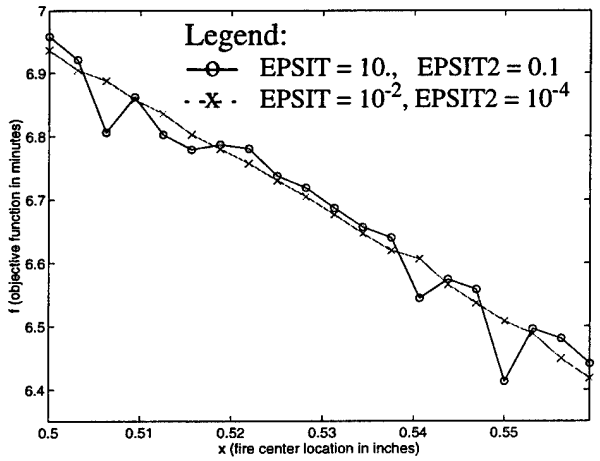


Figure 13. Detail of Figure 11 showing effect of QTRAN convergence tolerances on design space nonsmoothness.

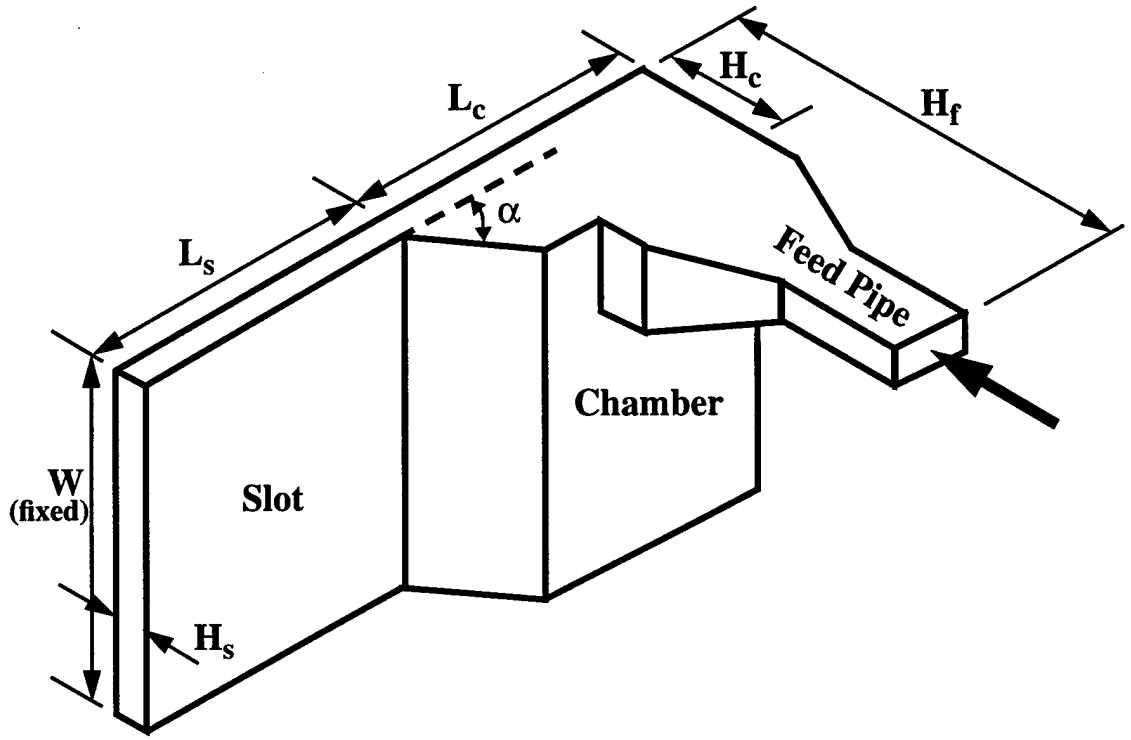


Figure 14. Schematic of a chamber-slot die.

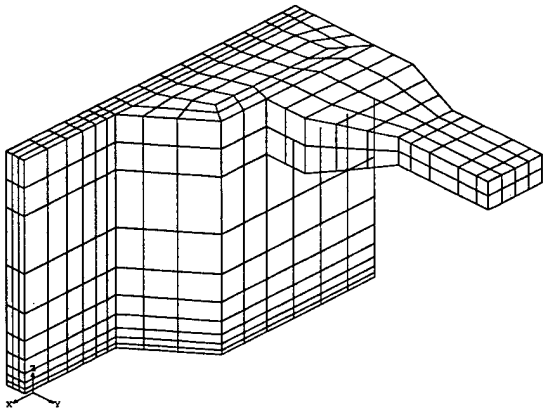


Figure 15. Initial geometry.

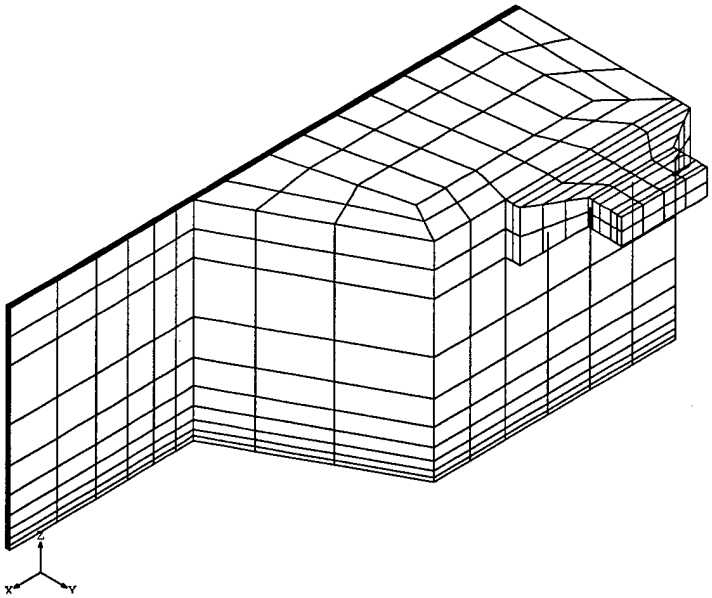


Figure 16. Final optimized geometry.

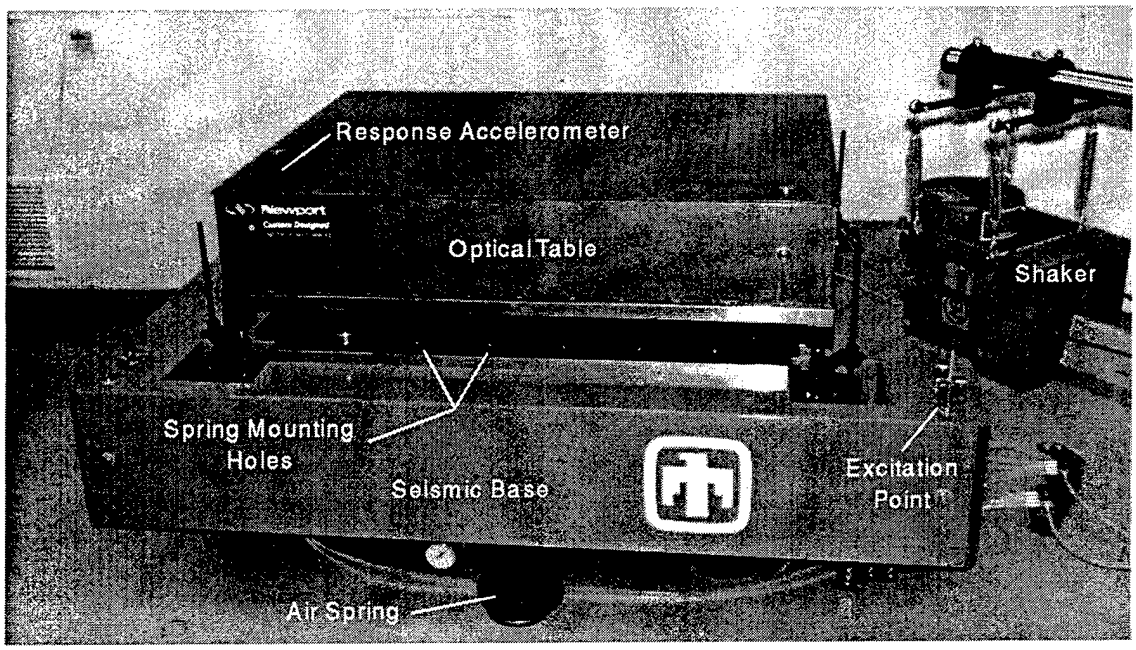


Figure 17. Vibration isolation testbed hardware.

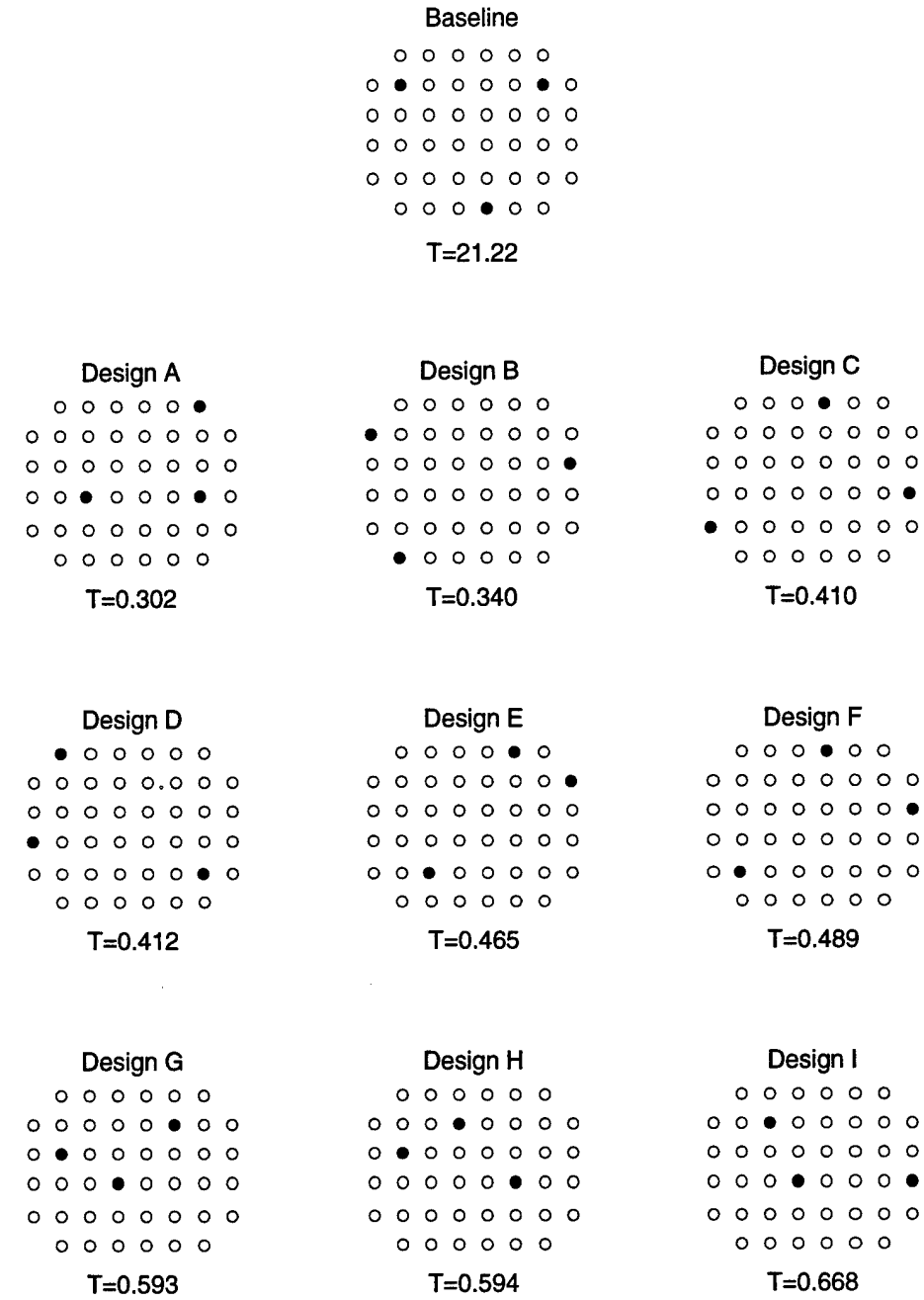


Figure 18. Spring locations and transmissibilities of baseline and 9 optimized designs.

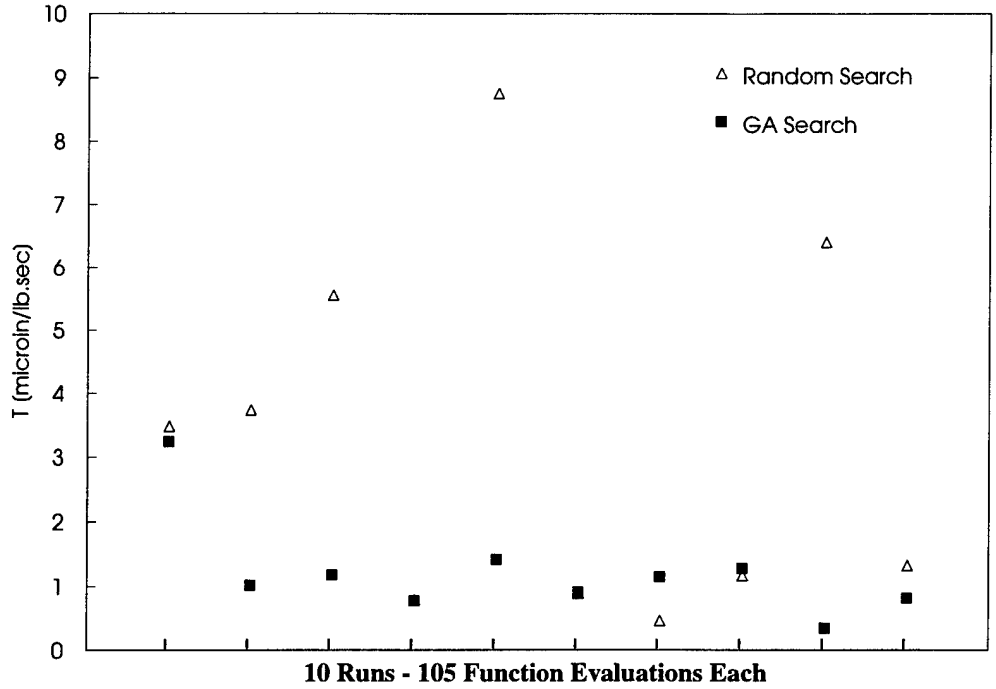


Figure 19. Results from 10 GA runs compared to 10 random searches for the same number of analyses.

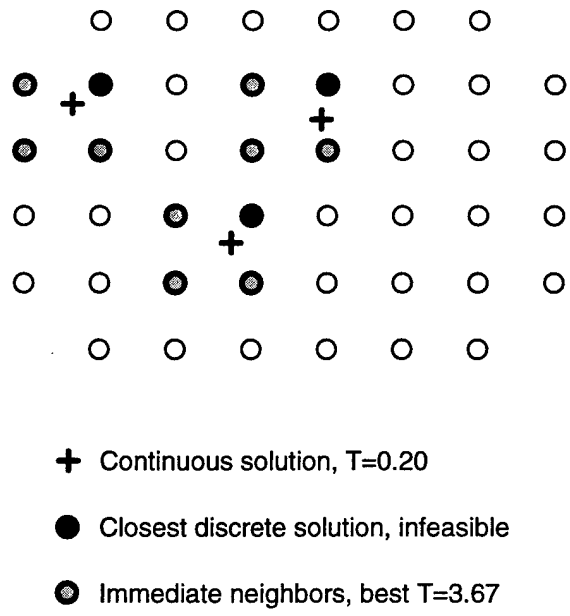


Figure 20. Comparison of continuous optimum and nearby discrete solutions.

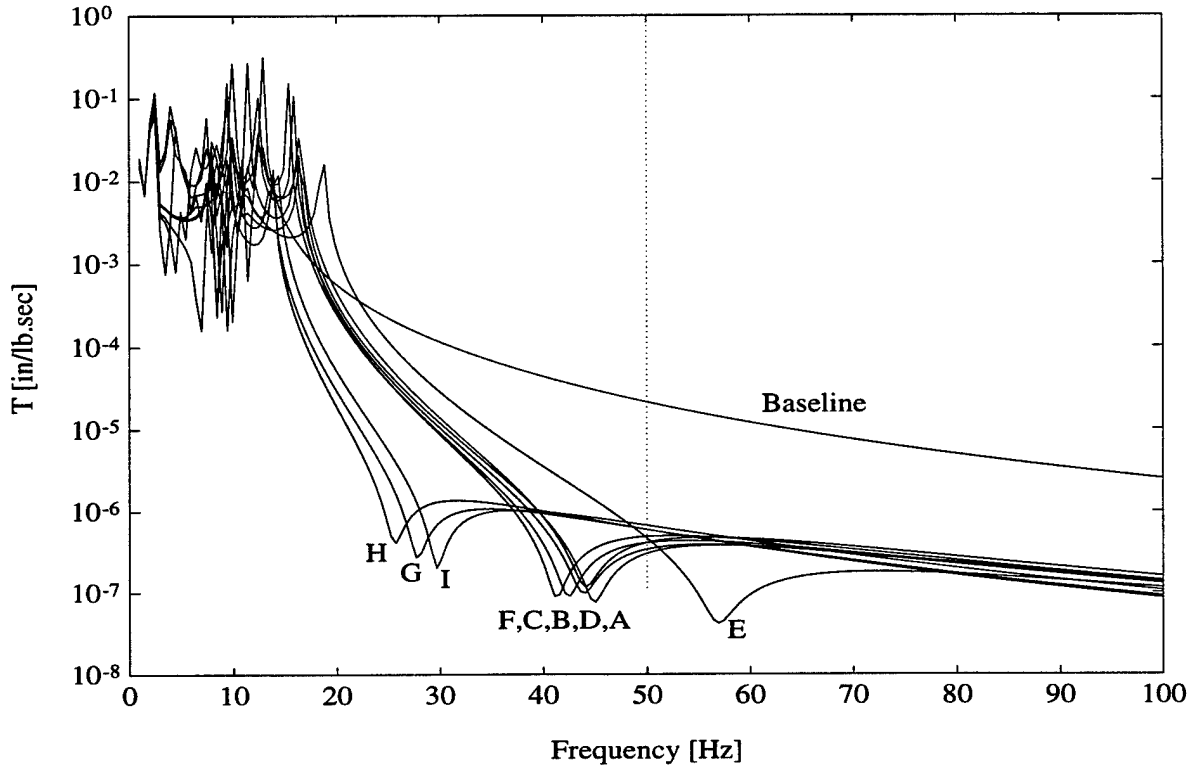


Figure 21. Frequency response functions for optimized designs.

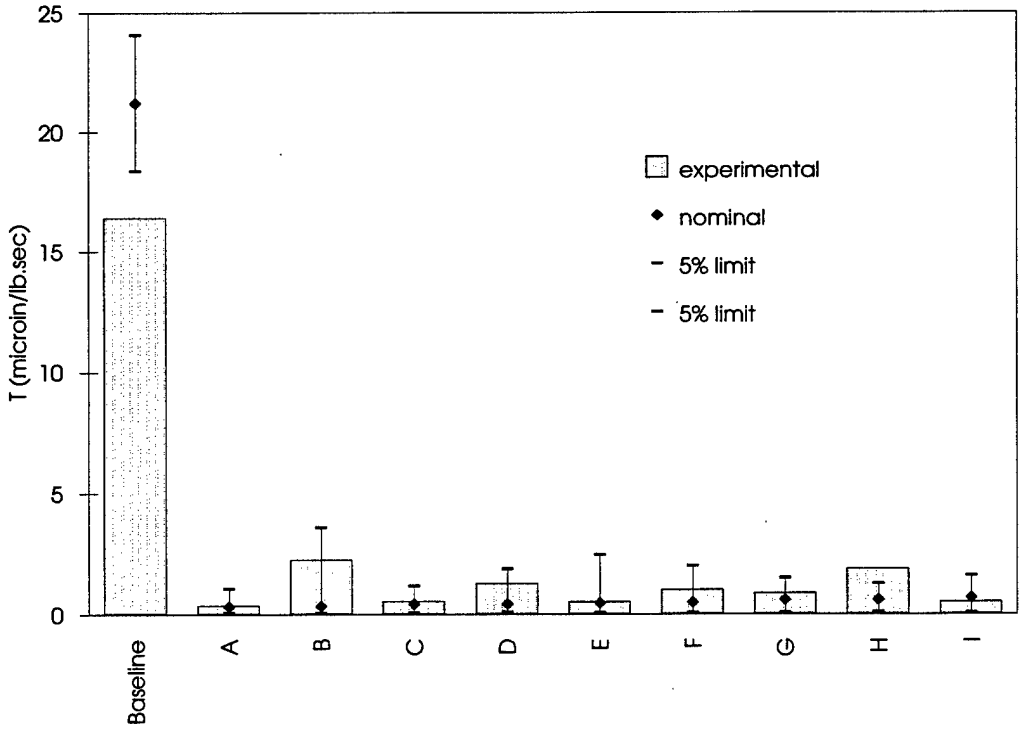


Figure 22. Comparison of analytical and experimental transmissibilities of baseline and optimized designs.

APPENDIX B

Utilizing Object-Oriented Design to Build Advanced Optimization Strategies with Generic Implementation

UTILIZING OBJECT-ORIENTED DESIGN TO BUILD ADVANCED OPTIMIZATION STRATEGIES WITH GENERIC IMPLEMENTATION

M.S. Eldred*, W.E. Hart†, W.J. Bohnhoff‡, V.J. Romero§, S.A. Hutchinson¶, and A.G. Salinger#

Sandia National Laboratories**
Albuquerque, NM 87185

Abstract

The benefits of applying optimization to computational models are well known, but their range of widespread application to date has been limited. This effort attempts to extend the disciplinary areas to which optimization algorithms may be readily applied through the development and application of advanced optimization strategies capable of handling the computational difficulties associated with complex simulation codes. Towards this goal, a flexible software framework is under continued development for the application of optimization techniques to broad classes of engineering applications, including those with high computational expense and nonsmooth, nonconvex design space features. Object-oriented software design with C++ has been employed as a tool in providing a flexible, extensible, and robust multidisciplinary toolkit that establishes the protocol for interfacing optimization with computationally-intensive simulations. In this paper, demonstrations of advanced optimization strategies using the software are presented in the hybridization and parallel processing research areas. Performance of the advanced strategies is compared with a benchmark nonlinear programming optimization.

Introduction

Computational methods developed in fluid mechanics, structural dynamics, heat transfer, nonlinear large-deformation mechanics, manufacturing and material processes, and many other fields of engineering can be an enormous aid to understanding the complex physical systems they simulate. Often, it is desired to utilize these simulations as virtual prototypes to improve or optimize the design of a particular system. The optimization effort at Sandia National Laboratories seeks to enhance the utility of this broad class of computational methods by enabling their use as design tools, so that simulations may be used not just for single-point predictions, but also for improving system performance in an automated fashion. System performance objectives can be formulated to minimize weight or defects or to maximize performance, reliability, throughput, reconfigurability, agility, or design robustness (insensitivity to off-nominal parameter values). A systematic, rapid method of determining these optimal solutions will lead to better designs and improved system performance and will reduce dependence on hardware and testing, which will shorten the design cycle and reduce development costs.

Towards these ends, this optimization effort has targeted the needs of a broad class of computational methods in order to provide a general optimization capability. Much work to date in the optimization community has focused on applying either gradient-based techniques to smooth, convex, potentially expensive problems[1] or global techniques to nonconvex but inexpensive problems[2]. When the difficulties of high computational expense and nonsmooth, nonconvex design spaces are coupled together, standard techniques may be ineffective and advanced strategies may be required. Moreover, since the challenges of each application are frequently very different, generality and flexibility of the advanced strategies are key concerns.

The coupling of optimization with complex computational methods is difficult, and optimization algorithms often fail to converge efficiently, if at all. The difficulties arise from the following traits, shared by many computational methods:

1. The time required to complete a single function eval-

*Senior Member of Technical Staff (SMTS), Structural Dynamics Dept., Mail Stop 0439, AIAA member.

†SMTS, Algorithms and Discrete Math Dept., Mail Stop 1110.

‡SMTS, Intelligent Systems Dept., Mail Stop 1177.

§SMTS, Thermal Sciences Dept., Mail Stop 0835.

¶SMTS, Parallel Computational Sciences Dept., Mail Stop 1111.

#Research Fellow, Parallel Computational Sciences Dept., Mail Stop 1111.

**P.O. Box 5800, Albuquerque, NM 87185, USA.
This work performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract DE-AC04-94AL85000.

uation with one parameter set is large. Hence, minimization of the number of function evaluations is vital.

2. Analytic derivatives (with respect to the parameters) of the objective and constraint functions are frequently unavailable. Hence, sensitivity-based optimization methods depend upon numerically generated gradients which require additional function evaluations for each scalar parameter.
3. The parameters may be either continuous or discrete, or a combination of the two.
4. The objective and constraint functions may not be smooth or well-behaved; i.e., the response surfaces can be severely nonlinear, discontinuous, or even undefined in some regions of the parameter space. The existence of several local extrema (multi-modality) is common.
5. Convergence tolerances in embedded iteration schemes introduce nonsmoothness (noise) in the function evaluation response surface, which can result in inaccurate numerical gradients.
6. Each function evaluation may require an "initial guess." Function evaluation dependence on the initial guess can cause additional nonsmoothness in the response surface. Moreover, a solution may not be attainable for an inadequate initial guess, which can restrict the size of the allowable parameter changes.

To be effective in addressing these technical issues, one must minimize the computational expense associated with repeated function evaluations (efficiency) and maximize the likelihood of successful navigation to the desired optimum (robustness). Important research areas for achieving these goals are fundamental algorithm research, algorithm hybridization, function approximation, parallel processing, and automatic differentiation. Research activities are ongoing in each of these areas at Sandia National Laboratories. The two research areas of central interest in this paper are:

Hybrid optimization techniques: The hybridization of optimization techniques exploits the strengths of different approaches and avoids their weaknesses. In a nonconvex design space, for example, one might initially employ a genetic algorithm to identify regions of high potential, and then switch to nonlinear programming techniques to quickly converge on the local extrema. Through hybridization, the optimization strategy can be tailored to suit the specific characteristics of a problem.

Parallel processing: The iterative nature of optimization lends itself to parallel computing environments. Since the simulation calls are independent for methods such as genetic algorithms and coordinate pattern search and for the finite difference gradient calculations of a nonlinear programming algorithm, parallelization can be achieved for single processor simulation codes by simultaneously executing many simulations, one per processor. Alternatively, parallel efficiencies can be gained through the interfacing of sequential optimization with parallel (i.e. multi-processor) simulations. More advanced strategies involve multi-level parallelism, in which parallel optimization strategies coordinate multiple simultaneous simulations of multi-processor codes.

Software Design

The DAKOTA (Design Analysis Kit for OpTimizAtion) toolkit utilizes object-oriented design with C++[3] to achieve a flexible, extensible interface between analysis codes and system-level iteration methods. This interface is intended to be very general, encompassing broad classes of numerical methods which have in common the need for repeated execution of simulation codes. The scope of iteration methods available in the DAKOTA system currently includes a variety of optimization, nondeterministic simulation, and parameter study methods. The breadth of algorithms reflects the belief that no one approach is a "silver bullet," in that different problems can have vastly different features making some approaches more amenable than others. Likewise, there is breadth in the analysis codes which may be interfaced. Currently, simulator programs in the disciplines of nonlinear solid mechanics, structural dynamics, fluid mechanics, and heat transfer have been utilized. The system, as will be demonstrated in this paper, also provides a platform for research and development of advanced methodologies.

Accomplishing the interface between analysis codes and iteration methods in a sufficiently general manner poses a difficult software design problem. These conceptual design issues are being resolved through the use of object-oriented programming techniques. In mating an iterator with an analysis code, generic interfaces have been built such that the individual specifics of each iterator and each analysis code are hidden. In this way, different iterator methods may be easily interchanged and different simulator programs may be quickly substituted without affecting the internal operation of the software. This isolation of complexity through the development of generic interfaces is a cornerstone of object-oriented design, and is required for the desired generality and flexibility of advanced strategies (e.g., hybrid algorithms and sequential approximate optimization).

The Application Interface (Figure 1) isolates application specifics from an iterator method by providing a generic interface for the mapping of a set of

parameters (e.g., a vector of design variables) into a set of responses (e.g., an objective function, constraints, and sensitivities). Housed within the Application Interface are three pieces of software. The input filter program ("IFilter") provides a communication link which transforms the set of input parameters into input files for the simulator program. The simulator program reads the input files and generates results in the form of output files or databases (a driver program/script is optional and is used to accomplish nontrivial command syntax and/or progress monitoring for adaptive simulation strategies). Finally, the output filter program ("OFilter") provides another communication link through the recovery of data from the output files and the computation of the desired response data set.

Optimizer iterators are part of a larger "iterator" hierarchy in the DAKOTA system. In addition to optimization algorithms, the DAKOTA system is designed to accommodate nondeterministic simulation and parameter study iterators. These three iterator classes frequently work together in a project: (1) parameter study is used to investigate local design space issues in order to help select the appropriate optimizer and optimizer controls, (2) optimization is used to find a best design, and (3) nondeterministic simulation is used to assess the affects of parameter uncertainty on the performance of the optimal design (a future extension will be to allow for optimization under conditions of uncertainty). Other classes of iterator methods may be added as they are envisioned, which "leverages" the utility of the Application Interface development. For example, software effort in coordinating multiple instances of parallel simulations on a massively parallel computer is reusable among all of the iterators in the DAKOTA system. The inheritance hierarchy of these iterators is shown in Figure 2. Inheritance enables direct hierarchical classification of iterators and exploits their commonality by limiting the individual coding which must be done to only those features which make each iterator unique.

Several optimization algorithm libraries and strategies are inherited from the **Optimizer** base class. DOT[4], NPSOL[5], OPT++[6], and SGOPT[7,8] have been incorporated in this framework as libraries of *stand-alone optimizers*. Additionally, the "Hybrid" and "SAO" optimization strategies are *combination strategies* which have been defined. In the Hybrid iterator, two or more stand-alone optimizers are combined in a hybrid strategy. Effective switching metrics are an important research issue. In the SAO iterator, stand-alone optimizers are interfaced with a separate function approximation toolbox in the setting of sequential approximate optimization[9] (SAO). Here, the accuracy and expense of the approximate subproblems, the mechanisms by which the approximations are updated, and the mechanisms of move limit enforcement are important research issues.

Application Descriptions

The breadth of application of the DAKOTA toolkit has been demonstrated previously in the disciplines of nonlinear solid mechanics, heat transfer, fluid mechanics, and structural dynamics[10]. These application investigations have uncovered challenges commonly encountered in real-world problems. It can be difficult to duplicate these challenges with suites of inexpensive test functions. Thus, the research investigations presented herein have focused on authentic engineering applications with the hopes that the performance observations will be pertinent to real-world problems, rather than merely being artifacts of the assumptions made in approximating with inexpensive test functions. For the purposes of demonstrating the advanced strategy developments, then, the focus will be placed on fire surety and chemical vapor deposition reactor applications.

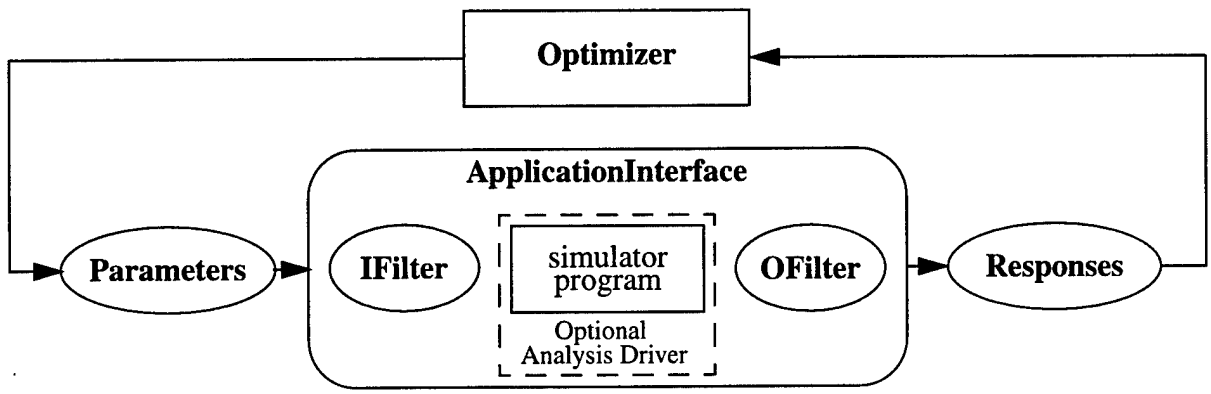


Figure 1. Application interface conceptualization.

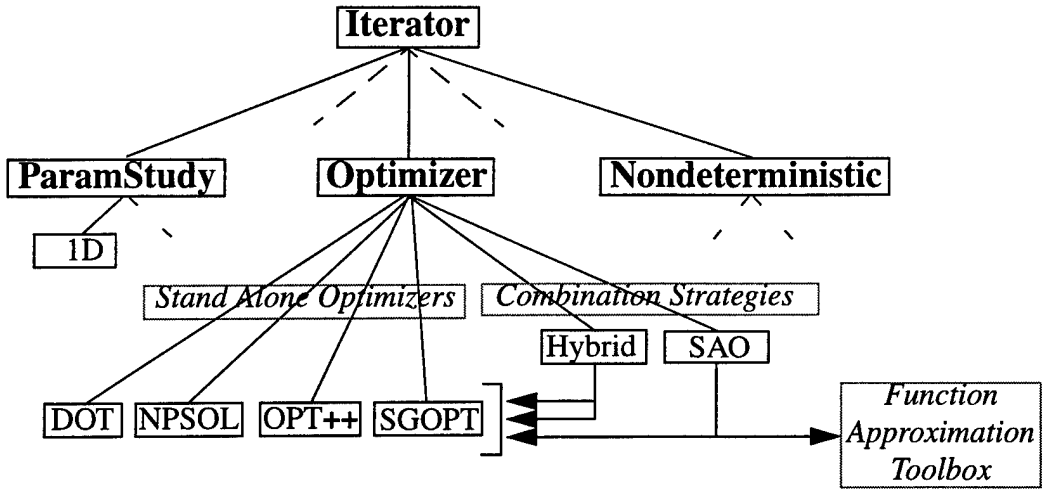


Figure 2. Iterator hierarchy showing broad iteration capabilities:
A) **ParamStudy**: for mapping response variations with respect to model parameters.
B) **Optimizer**: for numerical optimization studies.
C) **Nondeterministic**: for assessing the effect of modeling uncertainties on responses.

Heat Transfer: Determination of Worst Case Fire Environments

Problem Description. In thermal science simulations, parameter sets are sought which produce worst-case credible fire environment(s) for which structures and systems (such as aircraft, weapons, or petrochemical processing plants) must be designed. These inverse problems can be solved within an optimization framework. In this application, optimization techniques have been applied to determine the vulnerability of a safing device to a "smart fire"[11]. The optimization parameters consist of the location and diameter of a circular spot fire impinging on the device. The temperature of the fire is constant, though the heat flux it imparts to the device varies in time and space coupled to the response of the device. Function evaluations involved transient simulations using a nonlinear QTRAN[12] thermal model with radiative and conductive heat transfer. The finite element model used in the analysis is shown with typical temperature contours in Figure 3. Each simulation required between 8 and 60 CPU minutes to solve on an IBM SP2 node, depending upon the error tolerance levels specified.

The components of the safing device must work together to prevent the device from operating except under the intended conditions. It is a weaklink/ stronglink design: the weaklink is designed to fail under adverse conditions, which renders a potential stronglink failure incapable of harm. The weaklink is a Mylar-and-foil capacitor winding mounted on the outside of the safing device and the stronglink is a stainless steel plate mounted inside a cavity and offset right of center as shown in Figure 3. The time lag between failure of the stronglink and failure of the weaklink is the safety margin for the device and varies with the fire exposure pattern on the device surface. Hence, to validate the design of the safing system, the worst-case fire exposure pattern is sought by using optimization to minimize this safety margin for selective exposure to a 1000° C black body heat source.

Typical critical node temperature histories are shown in Figure 4 for a 20 hour fire exposure, where the critical nodes of the weaklink and stronglink are those which reach their failure temperatures earliest. The safety margin shown graphically is the objective function that the optimizer minimizes with respect to the design parameters of fire spot-radius (r) and fire center location (x), subject to simple bounds ($0.5 \le r \le 5.8$, $-2.9 \le x \le 2.9$). Early optimization studies solved this 2 parameter problem; later studies added the y degree of freedom in fire location as a third parameter ($0.0 \le y \le 2.9$). The specific techniques used in input filtering, adaptive simulation termination, and output filtering are discussed in a separate paper[10].

2-Parameter Optimization Results. This application is challenging due to the nonsmoothness and nonconvexity of the design space. In Figures 5 and 6, one-dimensional parameter studies show evidence of multimodality (Figure 6) and of slope-discontinuity at the minimum (Figures 5 and 6). The slope-discontinuity in the figures is caused by switching of the critical weaklink node between geometric extremes. Figure 5 shows negative curvature near the discontinuity (nonconvexity), whereas the discontinuity in Figure 6 is

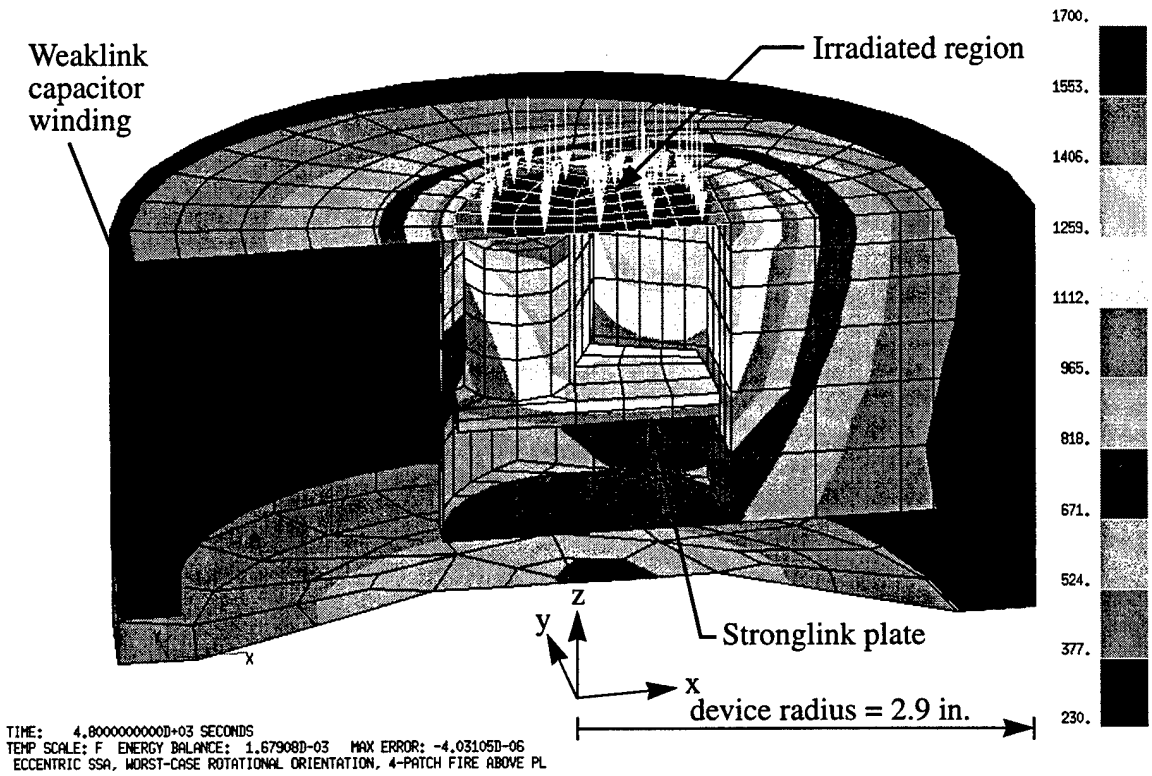


Figure 3. Finite element model and typical temperature distribution (^{o}F).

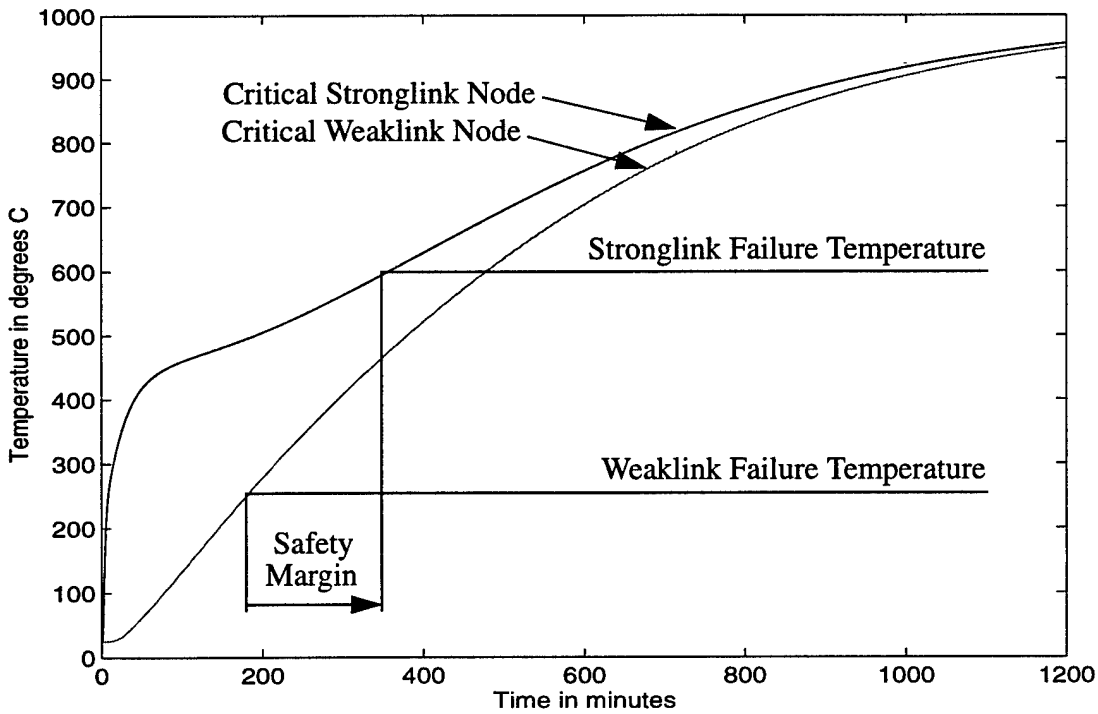


Figure 4. Temperature histories of critical stronglink and weaklink nodes.

more difficult to discern due to the positive curvature near the discontinuity.

These two parameter studies also provide insight into the mechanics of the problem. In Figure 5, the offset of the lowest safety margin from x=0 (the center of the device) backs up engineering intuition in that the stronglink is also offset right of center. That is, a fire centered roughly over the stronglink preferentially heats the stronglink and causes a lower safety margin. Figure 6 shows a less intuitive result, in which it is evident that the lowest safety margin is not achieved with either a large fire or a small fire. Rather, there exists an insidious, medium sized fire which is not so small that the heating rate is insufficient and which is not so large that it prevents selective heating.

Figure 7 is a detail of Figure 5 and shows evidence of small-scale nonsmoothness, which was reduced through the tightening of QTRAN convergence tolerances at the cost of approximately an order of magnitude greater computational time per analysis. EPSIT and EPSIT2 are absolute convergence tolerances in degrees which govern time step completion and node inclusion in nonlinear iterations, respectively. The additional computational expense per analysis was warranted in this case since none of the nonlinear programming algorithms could successfully navigate the design space without reducing the nonsmoothness (even with large finite difference step sizes).

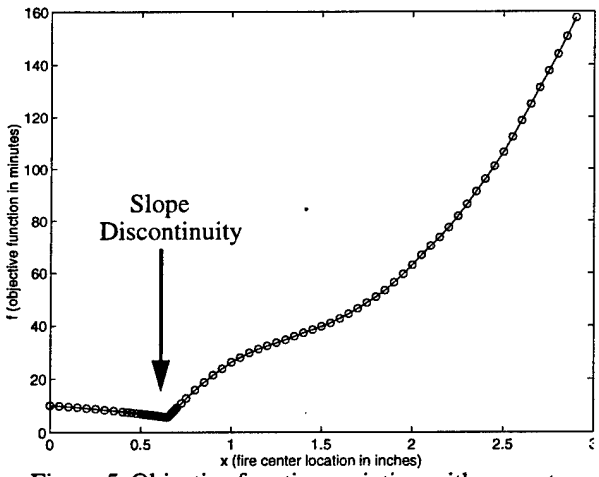


Figure 5. Objective function variation with respect to fire center location (x) for r=1.89.

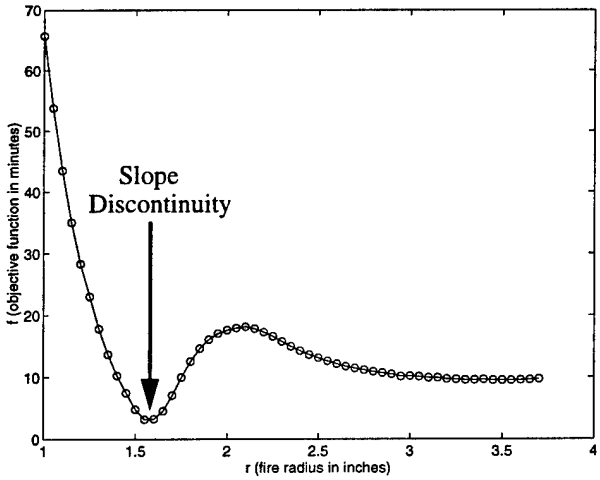


Figure 6. Objective function variation with respect to fire radius (r) for x=0.8.

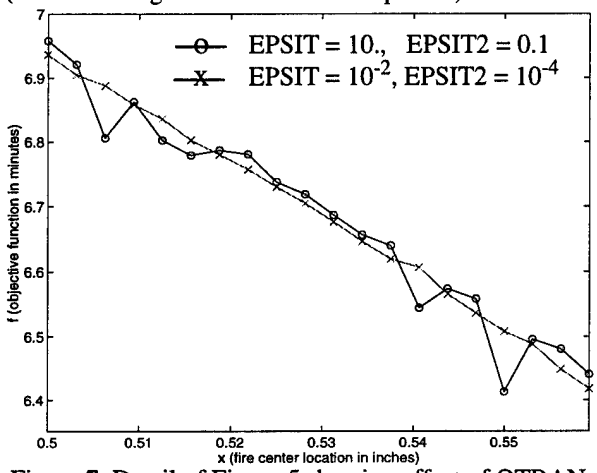


Figure 7. Detail of Figure 5 showing effect of QTRAN convergence tolerances on design space nonsmoothness.

Performance comparisons of nonlinear programming (NLP) algorithms are detailed in a previous paper[10]. In summary, Newton-based optimizers performed poorly due to the nonconvexity of the design space (a quadratic approximation is a poor representation); conjugate gradient (CG) methods were much more successful. Choice of finite difference step size (FDSS) for computation of numerical gradients proved to be important. FDSS should be as small as possible to allow for effective convergence to a minimum, but still large enough that small-scale nonsmoothness does not cause erroneous gradients. Lastly, for nonsmooth applications, a robust line search (as opposed to an aggressive search tuned for smooth applications) was shown to be essential in enabling reliable navigation to the optimum from different starting points. The lowest objective function value found for the 2 parameter problem was 2.531 minutes (r=1.620, x=0.7820) at tight tolerances (EPSIT=10^{-4}, EPSIT2=10^{-6}) which, when compared to stronglink and weaklink failure times of 62.743 and 60.212 minutes respectively, corresponds to a safety margin of just 4%.

3-Parameter Optimization Results. In more recent studies, the fire parameterization was extended to 3 parameters (y degree of freedom in fire location added) in order to investigate if fires centered off the line of symmetry (see Figure 3) could result in lower safety

margins. It should be noted that fires centered off the line of symmetry (i.e., y≠0) are mirrored by the symmetry condition; that is, there is an identical fire exposure in the y<0 half-plane. A typical parameter study for objective function variation with respect to y is shown in Figure 8. This study shows that the addition of the y parameter is unlikely to result in additional reduction of the safety margin. This is intuitive since fires located off the symmetry line will not be as successful in preferentially heating the stronglink. The detail insert shows additional small-scale nonsmoothness in the vicinity of y=0. In contrast with Figure 7, tightening the EPSIT tolerances does not result in significant smoothness improvements. Thus, this nonsmoothness is believed to be geometry related and not an artifact of finite numerics.

For the 3 parameter problem, performance of coordinate pattern search (CPS) optimizers from the SGOPT package has been compared with that of NLP (DOT's Fletcher-Reeves CG). Figure 9 shows the optimization wall clock history for serial CPS and two NLP studies. The two NLP studies both employ 0.1% FDSS, but differ in the EPSIT tolerances employed in the simulations. For (10^{-2}, 10^{-4}) EPSIT tolerances, NLP terminates prematurely and the 2.5 minute minimum safety margin is never reached. This is a clear indication of smoothness levels which are insufficient to allow effective navigation of the optimizer for the chosen FDSS. At these tolerance levels, each simulation requires approximately 20 CPU minutes to solve. Tightening the EPSIT tolerances to the (10^{-4}, 10^{-6}) level increases the individual simulation expense to approximately 60 CPU minutes, but allows for effective navigation to a 2.537 minute safety margin in 96 wall-clock hours on a dedicated machine (this 3-parameter result is slightly less optimal than the 2.531 minute 2-parameter result because the same level of optimization convergence was not enforced).

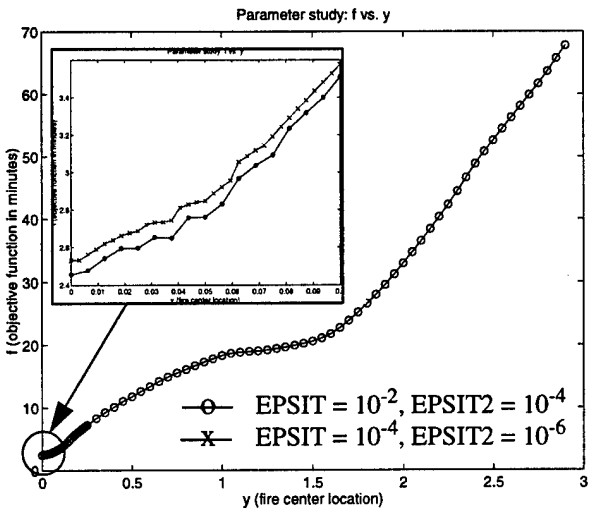


Figure 8. Objective function variation with respect to fire location (y) for r=1.6204 and x=0.78205. Detail shows local nonsmoothness for 0≤y≤0.1

Since CPS is less sensitive to small-scale nonsmoothness than gradient-based techniques, looser EPSIT tolerances can be employed which lowers the individual fire simulation expense considerably, to approximately 8 CPU minutes each. Figure 9 shows rapid convergence of CPS to the vicinity of the minimum and final convergence to a safety margin of 2.504 minutes in 28.5 wall-clock hours. While the number of function evaluations required by CPS is greater than gradient-based optimization (220 compared to 96 in Figure 9), the lower individual simulation expense more than compensates, making the overall computational expense of the CPS optimization more than 3 times lower than that of the gradient-based optimization.

However, evaluation of the CPS optimal point with tight tolerances (EPSIT=10^{-4}, EPSIT2=10^{-6}) reveals a tight-tolerance safety margin of 2.649 minutes, which is 4.4% less optimal than the NLP result. This highlights the weakness of using CPS with inexpensive function evaluations: convergence is not as exact. This is intuitive since, as CPS progresses towards convergence, the step size decreases and the substantial nonsmoothness present with loose tolerances becomes more of a hindrance. If the NLP optimization was terminated when this level of optimality was achieved, the NLP run time reduces to 73.3 wall-clock hours and the efficiency gains measured with CPS reduce accordingly.

Global optimization issues have also been investigated with this application. The studies in Figure 9 started from a good initial guess of (r,x,y,)=(1.4, 0.5,

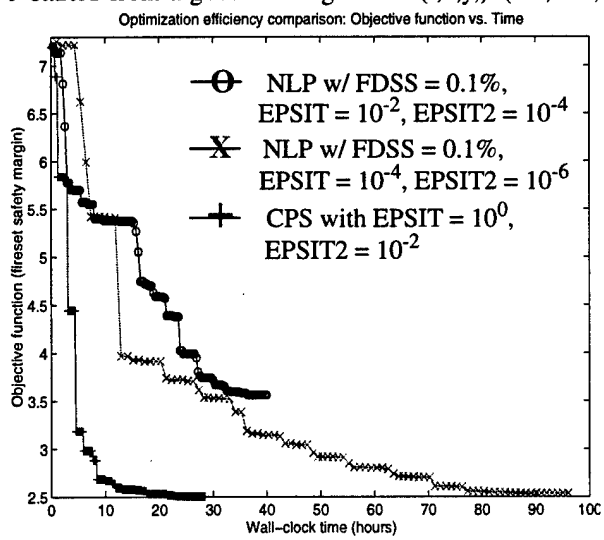


Figure 9. Optimization history comparison: Best objective function value vs. wall-clock time in hours

0.0) with an initial objective function of 7.25 minutes. Starting from a different initial guess of (1.9, 2.1, 0.0) with an initial objective function of 69.4 minutes, Figure 10 shows the relative performance of CPS, NLP, and an exploratory real-valued GA. Two shorter GA runs of 15 generations each were performed rather than one long 30 generation search, because some research indicates that several short searches usually outperform a single long search[13]. The 2 GA studies differ only in the initial random population seed and the best run of the two is shown in the Figure. The GA can employ very loose tolerances (EPSIT = 10^1, EPSIT2 = 10^{-1}), and the selected settings are nonaveraging 2-point crossover, population size of 15, elitist retention of the 2 best individuals in the population, and uniform mutation at a 40% rate. This is a difficult problem for the GA, since the size of the region containing the 2.5 minute global minimum safety margin is a relatively small portion of the total design space. A Monte Carlo simulation of 120 random points (not shown) found only 2 fires with safety margins lower than the "big fire" safety margin of approximately 10 minutes, and both of these points were only slightly better with objective functions of 7.45 and 7.76 minutes. Thus, finding the global minimum region in an initial population of a GA is unlikely, and the GA must rely primarily on mutation to search for this region. In addition, the region is small and steep and the topography in that vicinity contains mostly large objective functions, and these unfit population members tend to push the GA population away from this region. Moreover, the GA is naturally attracted to the "big fire" solution, since this solution makes up a large portion of the design space and its objective function of approximately 10 minutes is a "strong base of attraction," meaning that the population members with

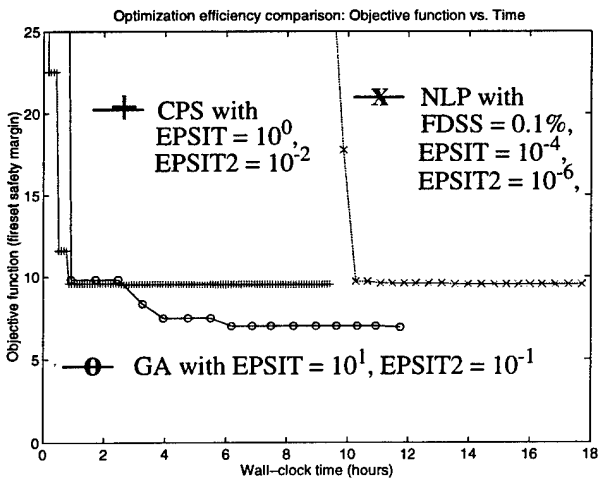


Figure 10. Optimization history for GA runs compared with CPS and NLP starting from (r,x,y)=(1.9, 2.1, 0.0).

these values are more fit than most other members. To combat this behavior, mutation was set at a relatively high 40% and the r upper bound was reduced to 2.9 (which still allows for a full face fire but restricts its dominance in the initial population). With these adjustments, it is evident in Figure 10 that the GA is better suited for handling multimodality than CPS or NLP and is successful in locating a promising region for local search. The CPS and NLP approaches both become trapped in the "big fire" local minimum with an objective function of approximately 10 minutes. The best GA solution of approximately 7 minutes will be used as the first pass in several hybridization studies.

The pertinent observations in efficiency comparisons between CPS, GAs, and NLP are summarized as:

- CPS can be an efficient alternative to NLP, especially if local design space smoothness is tied to simulation expense, since CPS is less sensitive to nonsmoothness and can navigate effectively using inexpensive simulations. NLP is better, however, at precise convergence. This points to potential in a hybrid CPS/NLP strategy in which CPS is used to "get close" and NLP provides final convergence to the precise minimum.
- Gradient-based optimizers put substantial faith in the accuracy of the computed search direction, and in nonsmooth applications, this level of faith may not be justified since the gradients used to calculate the search direction have questionable accuracy. CPS optimizers do not confine themselves to a single search direction, but rather search multiple directions simultaneously. As a result, they can be more robust in nonsmooth applications. Furthermore, these multiple searches are independent, which provides easily-exploitable coarse-grained parallelism. CPS is, however, susceptible to the "curse of dimensionality," meaning that the method is most competitive in efficiency when the number of design variables is small.
- Genetic algorithms are good techniques for global design space feature extraction and location of promising regions for refined searches. Since they are zero-order techniques, inexpensive models may be used for the evaluations. In addition, they have very exploitable parallelism since each evaluation in a population cycle is entirely independent. However, GAs are not infallible. For problems with isolated minima lacking exploitable design space structure, Monte Carlo sampling or grid search may be the most effective global identification approach.

Chemically Reacting Flows: CVD Reactor Design

Problem Description. There are many choices associated with the design and operation of Chemical Vapor Deposition (CVD) reactors including geometry, inlet concentrations, temperature and velocities, disk temperature and spin rate, etc. Ultimately, one wishes to maximize profit in the deposition process by producing crystals with a high degree of uniformity and purity at the lowest possible manufacturing costs. Since building and experimenting with actual reactors is an expensive ($1000/hour) and often hazardous process, virtual prototyping through the use of numerical simulations and optimization techniques can help reduce the cost and risk associated with reactor design.

A horizontal CVD reactor for the growth of Gallium Arsenide (GaAs) from arsine and trimethylgallium (TMG)[14] is of interest in this study (Figure 11). The reactants flow through a horizontal vessel with a tilted base which is heated to the temperature at which deposition occurs. In the middle of the heated region is a rotating disk on which uniform growth can occur. Figure 11 also illustrates the simulation of the reactor using MPSalsa, a chemically reacting flow finite-element code developed at Sandia National Laboratories for use on Massively Parallel (MP) MIMD computers[15,16], by showing the path of fluid through the reactor and asymmetric surface contours of the main reactant. Figure 12 shows a deposition profile of GaAs on the reactive surface where the circular outline is the spinning disk boundary. Also in Figure 12 is a graph of the spin-averaged deposition on the disk as a function of radial position. Ideal operation of the reactor would consist of an average growth rate of 10-20 Angstroms/second and a perfectly uniform deposition profile.

As reactors and processes can vary, so do the relevant design parameters. For this problem, we have chosen to optimize an objective function which includes the operating and material costs of the reactor less the gain in value of the resulting wafer. A quadratic penalty term is added to restrict the growth rate from too large a value, which would lead to poor crystal quality. This objective function models some of the trade-offs faced by reactor operators: growth rate vs. product uniformity and materials costs vs. growth rate. It has units of $/hour and takes into account both costs and revenue, so that the further negative the objective function value, the more profit the process is making. An optimal configuration is sought by varying 3 operating parameters: the inlet concentration of trimethylgallium, the inlet flow rate, and the rotation rate of the reacting disk. The capability of performing geometric optimization has recently

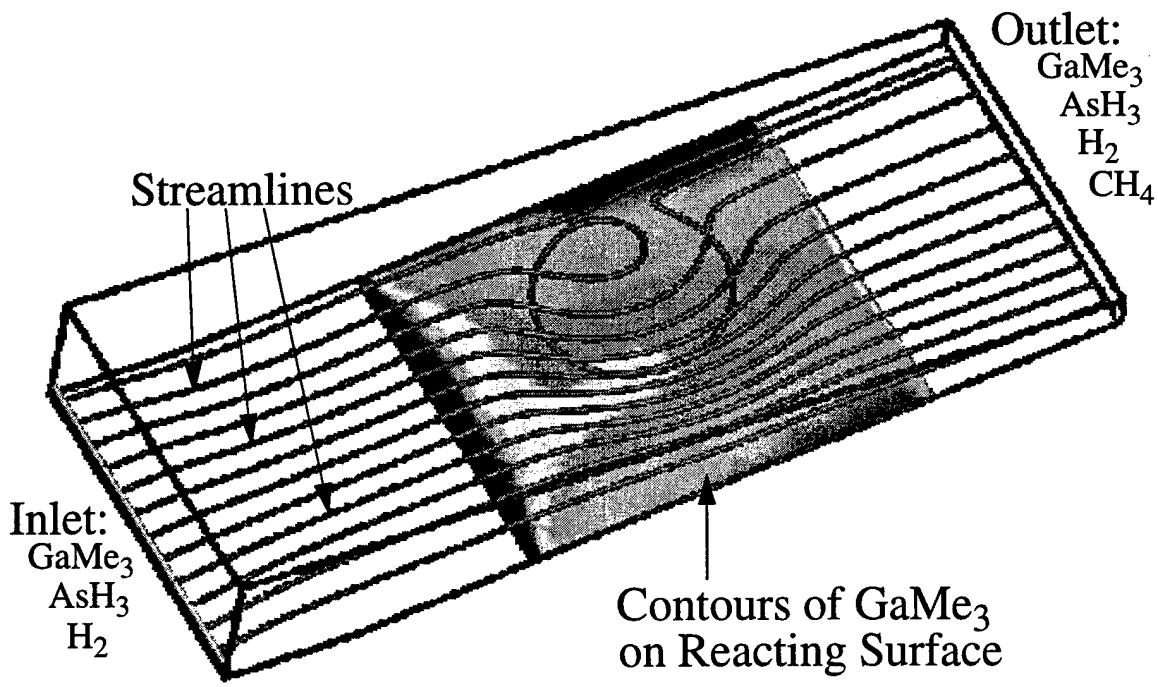


Figure 11. Deposition of Gallium Arsenide in a horizontal CVD reactor with tilted susceptor.

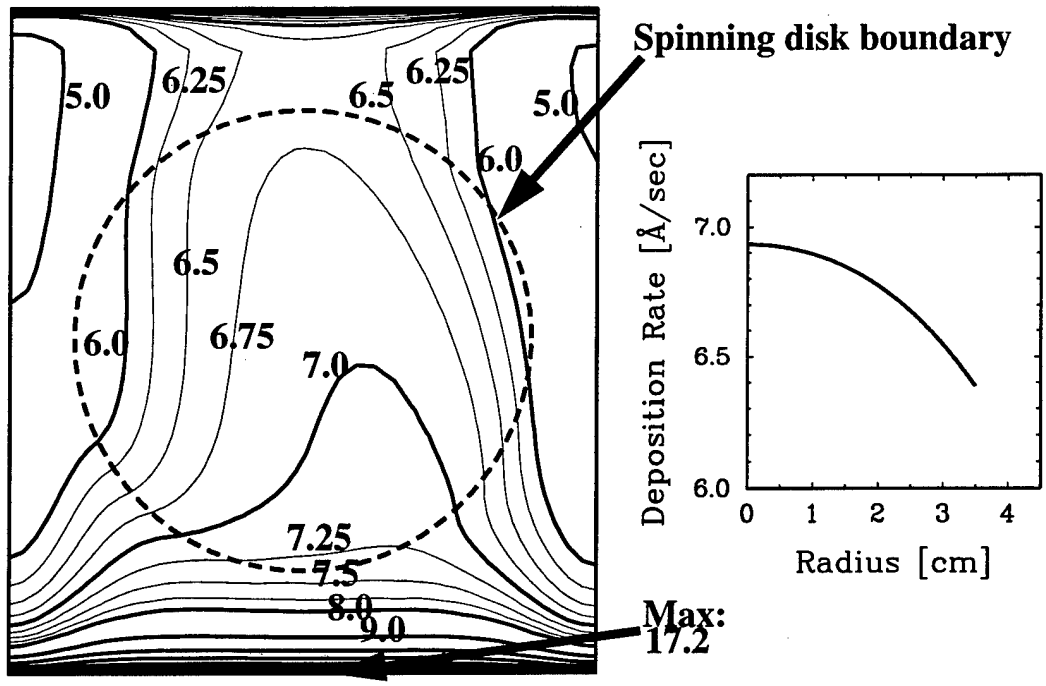


Figure 12. Contours of the deposition rate of GaAs over the entire reactive surface (left) and time-averaged deposition rate over the spinning disk (right). Attempting to maximize the growth rate while maintaining required uniformity over the disk is a major component of the optimization of this reactor.

been added by representing the tilt-angle of the reactor as an additional parameter.

Optimization Results. For this problem, two finite element meshes of the reactor geometry (Figure 11) were used: a coarse mesh for quick investigation of the parameter space and verification of the methodology, and a fine mesh for more accurate results. This was done to expedite the optimization process and to make the best use of the Paragon resources. The coarse mesh was comprised of 8504 hexahedral elements and 10188 nodes while the fine mesh had 36720 hexahedral elements and 40720 nodes. At each node in the mesh there are 9 unknowns (three velocity component, pressure, temperature and four species mass-fractions) resulting in a total problem sizes of 91692 and 366480 unknowns, respectively, for the coarse and fine meshes. The coarse mesh has been used as an approximation by first finding an optimum on the coarse mesh and then using these parameter values to start the more expensive fine mesh calculations.

The first coarse mesh run optimized the solution over 3 operating parameters: the inlet reactant concentration, the inlet flow velocity, and the disk spin rate. DOT's conjugate gradient algorithm was selected as the optimizer. After 34 function evaluations (including finite difference gradient calculations) and 4 iterations of the conjugate gradient technique, an optimum was found. Figure 13 shows the value of the objective function for each function evaluation.

A second optimization run on the coarse mesh was performed by adding in the tilt-angle of the reactor base as a fourth parameter. This run started at the optimum from the previous run, and decreased the objective function a little further from -1178.8 to -1226.0, while

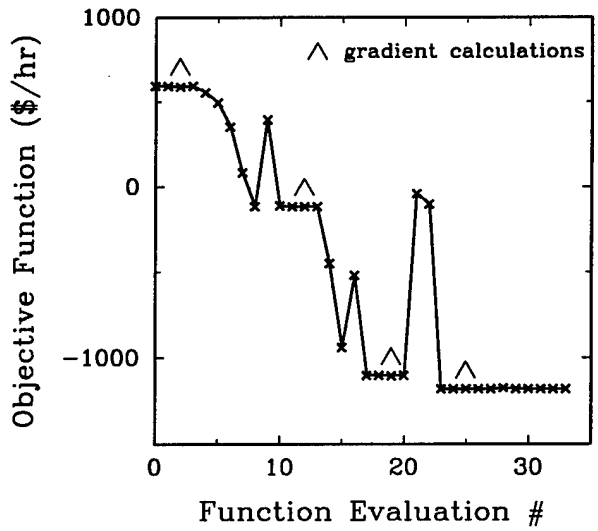


Figure 13. Objective function history for a 3 parameter optimization of a CVD reactor on a coarse mesh. Each conjugate gradient iteration began with a gradient calculation as marked.

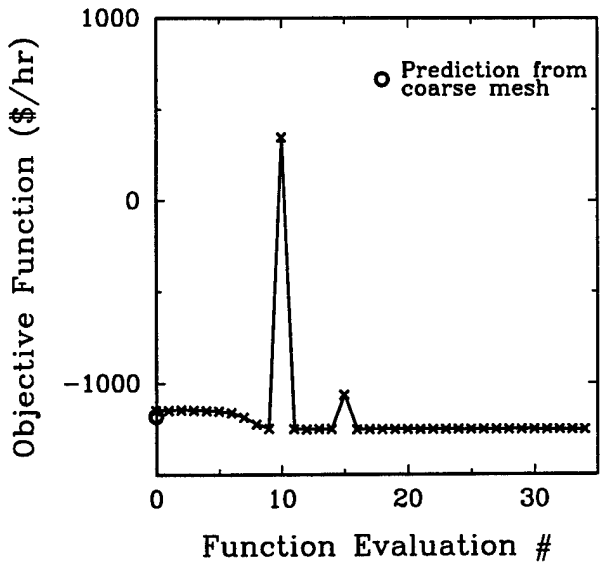


Figure 14. Objective function history for CVD reactor optimization on the fine mesh with initial guess from the coarse mesh converged solution. Each function evaluation required the solution of 366480 unknowns, which were solved on 512 processors of the Intel Paragon in 5-8 minutes each.

increasing the tilt angle from 9 degrees up to 11, which was set as the upper bound.

Finally, a 3-parameter optimization run using the fine mesh was initiated with the optimal parameter values from the first coarse mesh run. As can be seen in Figure 14, the optimization run converged after 35 function evaluations, although it was nearly converged much sooner. The objective function decreased from -1144.0 (corresponding to the coarse mesh objective of -1178.8) down to -1250.1. The initial guess provided by the coarse mesh proved to be a good approximation to the fine mesh, as one parameter changed by about 10% while the others were less than 2% from the optimum. The use of a coarse mesh to rapidly identify promising areas of parameter space for more expensive fine mesh runs can be an important resource-saving methodology.

Parallel Processing

Strategy Discussion. High performance computing is an essential technology in optimization research. For GAs, CPS methods, and the finite difference gradient calculations of an NLP algorithm, many simulations are entirely independent, making it possible to achieve "embarrassingly parallel" strategies using a coarse-grained approach (i.e., simultaneously executing many single-processor simulations, one per node). The other attractive location for parallelism is in the simulation code itself, and Sandia has been a leader in developing massively parallel (MP) simulation capabilities. Thus, a second approach to parallel efficiency is that of mating an efficient, sequential optimizer (i.e. NLP) with an MP simulation code.

Two parallel optimization studies have been performed. The fireset application uses a parallel optimization algorithm which invokes multiple independent simulations of single-processor codes. The CVD reactor design study achieves parallel efficiency through sequential optimization with MP simulation codes.

Heat Transfer: Determination of Worst Case Fire Environments

Parallel CPS from the SGOPT package has been used for improving efficiency in optimization of fire surety simulations. Individual thermal simulations execute on nodes of the IBM SP2 using the native loadleveler software to select lightly loaded nodes, and multiple simulations execute simultaneously.

CPS executes 2 simulations in each of n parameter directions during an iteration. The end of an iteration is a synchronization point for the parallel algorithm; thus, $2n$ simulations at most may be performed in parallel. Then, the maximum possible parallel speedup for the 3 parameter fire surety application using single-processor analyses is 6. In practice, observed parallel speedup was limited by the availability of only 3 commercial QTRAN licenses.

Figure 15 shows the optimization wall clock history for serial CPS, parallel CPS limited by 3 commercial QTRAN licenses (observed performance), and parallel CPS with unlimited QTRAN licenses (potential performance as limited by algorithm rather than by licenses). Reductions in wall-clock time of a factor of 10 for the CPS optimization are observed over that of the NLP

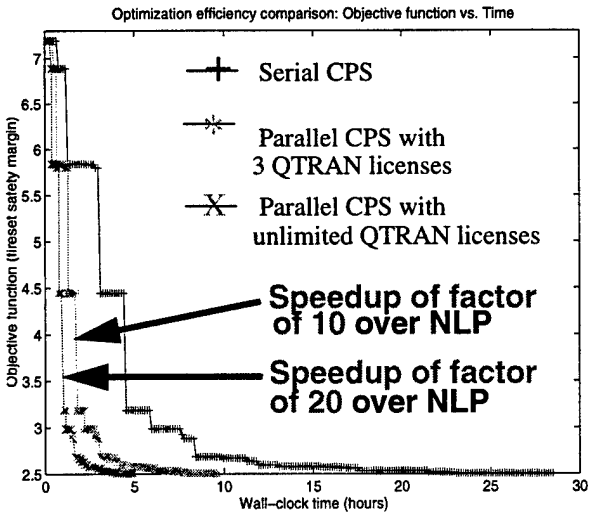


Figure 15. Optimization history comparison: Best objective function value vs. wall-clock time in hours

optimization (see Figure 9), and a factor of 20 savings would be possible with additional QTRAN licenses. As stated previously, however, the CPS results are slightly less optimal, from which the utility of a CPS/NLP hybrid can be inferred.

Parallel genetic algorithms are also under investigation (results not available at time of printing). More parallelism is possible with GAs than with CPS in this problem, since the number of possible simultaneous analyses for the GA is determined by population size (15 used in Figure 10), compared with 6 possible simultaneous analyses ($2n$ where $n = 3$) for parallel CPS. In addition, some research suggests that employing multiple independent GA populations in parallel can be an effective technique, and this removes this population size speedup limit. However, in this problem, observed performance will still be limited by the 3 available licenses.

Chemically Reacting Flows: CVD Reactor Design

Massively parallel simulations have been employed in NLP optimization studies to allow for expeditious analysis of high fidelity models of the chemically reacting flows within a CVD reactor. MP SALSA simulations execute on a partition of nodes on Sandia's 1840 node Intel Paragon.

For each set of parameters given by DAKOTA (either 3 or 4 parameters), a steady state problem is solved by MPSalsa from which a single objective function is calculated. Each function evaluation on the coarse mesh takes 2-3 minutes on 256 Intel Paragon processors, while a function evaluation on the fine mesh requires 5-8 minutes on 512 processors. For each problem size, there is a trade-off between computational speed-up and interprocessor communication overhead and these numbers of processors achieve an effective balance for these problem sizes.

Through the use of the MP computer, the relatively short objective function evaluation times are enabling us to optimize this and larger design problems of interest to the CVD processing industry. The total CPU time used for the coarse mesh optimization study was approximately 80 minutes on each of 256 Paragon processors, and the total CPU time for the fine mesh optimization study was around 6 hours on each of 512 Paragon processors, including time for I/O.

Code Modifications and Operation. An input filter script has been generated to control and pass information to the MPSalsa program. Also, MPSalsa has been modified to take this information from DAKOTA and generate the returned objective function value. This modification also allows MPSalsa to stay resident on the parallel machines and thus alleviates the need for re-initialization for every objective function evaluation. Further, MPSalsa can use the previous solution as an initial guess for each evaluation. With these two time saving measures, we have cut the total CPU time by a factor of 2-10 (depending largely on the mesh size) over the option of re-launching MPSalsa for every function evaluation. Since the SUNMOS operating system on the Paragon only supports a single process at a time, only MPSalsa will be run on the Paragon. Thus, the DAKOTA and the filter script communicate (from a front-end machine) with MPSalsa via parameter and other control files on a common disk system.

Observations. Through the use of massively parallel computing, accurate simulation of complicated engineering systems such as CVD reactors is possible and relatively rapid. With this capability comes the opportunity to use optimization algorithms to locate improvements in operation and design. As a proof-of-concept, optimal values of three key operating parameters have been located for the CVD growth of Gallium Arsenide semiconductor crystals, with respect to an objective function that takes into account materials costs, growth rate, and the uniformity of deposition. The resulting solution was better than any previously simulated and is believed to be a global optimum. Initial simulations adding in a fourth design parameter have already shown that changes in the reactor configuration can be made to improve the profitability of the reactor. It has been shown that using a coarse mesh for initial optimization studies can efficiently locate promising areas of parameter space for the accurate fine mesh.

Given the heavy use Sandia's MP computers receive, it is imperative that efficient use is made of these resources. To this end, it is planned to augment the DAKOTA/MPSalsa scheme in order to provide a two-level parallelization scheme. This would allow independent objective function calculations to be done concurrently, even while these calculations are themselves parallel. Typically, the most efficient number of processor on which to run a problem is the minimum required (owing to communication costs). Thus, by evaluating the objective functions on the minimum number of processors and by performing several of these in parallel, on can achieve nearly linear speedup and optimal efficiency as shown in [17]. For gradient methods, this second level of parallelization is limited to the number of optimization parameters (within a finite difference gradient calculation) but will remain more effective than simply increasing the number of processors used for a particular objective function solution.

Algorithm Hybridization

Strategy Discussion. Hybrid optimization algorithms seek to enhance the overall robustness and efficiency of an optimization approach by tailoring algorithm strengths to different parts of the optimization process. For example, in an optimization problem whose design space may contain multiple minima, the initial stages of an optimization process should be characterized by an identification of promising design space regions. Algorithms suited for this (e.g., genetic algorithms) are often expensive since they usually require many function evaluations. Thus, these algorithms should only be used long enough to serve their identification purpose. Once promising regions have been located, an efficient local technique (e.g., NLP) can be used to converge on precise minima. An important associated technique is that of variable complexity modeling[18], in which analysis "complexity" (e.g., mesh density, convergence criteria) is tailored to meet the needs of the current algorithm or optimization phase. In the example cited above, it is clearly attractive to use loose convergence tolerances in the initial identification phase (since a genetic algorithm approach does not require smooth differentiability of the response surface), followed by appropriately tight tolerances in the local convergence phase.

Global/local hybrids are not the only example. It has been shown previously that CPS and NLP have differing performance in the presence of local nonsmoothness. Thus, an efficient local strategy would combine CPS using inexpensive function evaluations in the initial optimization phase with NLP using expensive evaluations in the final convergence phase.

An important point of research is the development of appropriate algorithm switching metrics. In the studies investigated below, the approach employed is that of staying with an algorithm as long as it is making progress. When an algorithm's progress slows or when it's function evaluation budget has been spent, the hybrid strategy switches to the next algorithm and continues.

Heat Transfer: Determination of Worst Case Fire Environments

GA/NLP and GA/CPS two-pass hybrids with variable complexity modeling. The GA initial phase uses inexpensive function evaluations (EPSIT = 10^1, EPSIT2 = 10^{-1}) to stochastically identify promising design space regions. As shown in Figure 10, the GA performs 15 population cycles and identifies a promising region with an objective functions of 6.930. In the hybridization study, the best point found after 10 population cycles

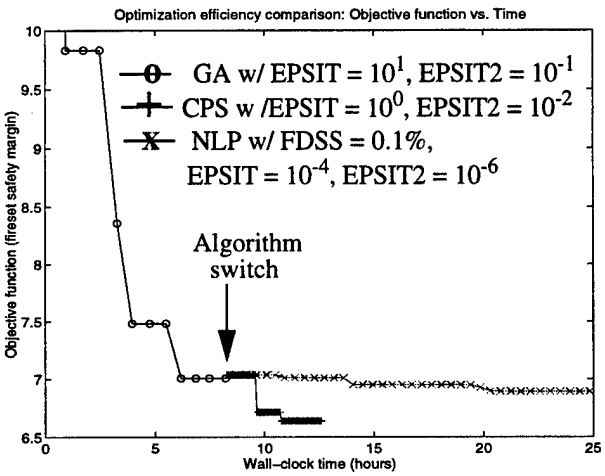


Figure 16. Optimization history comparison for hybrid GA/NLP and GA/CPS strategies: Best objective function value vs. wall-clock time in hours

(7.005 minutes) is handed off to CPS and NLP approaches for local convergence. The CPS second phase uses slightly tighter tolerance evaluations (EPSIT = 10^0, EPSIT2 = 10^{-2}), whereas the NLP second phase requires tight tolerance, expensive evaluations (EPSIT = 10^{-4}, EPSIT2 = 10^{-6}). Figure 16 shows the optimization history and relative performance of the GA/NLP and GA/CPS hybrids. It is evident that the starting point for the CPS and NLP second phase studies is not sufficiently close to the global minimum since both approaches become trapped in a local minimum with an objective function slightly less than 7 minutes. The GA/CPS hybrid converges on this local minimum in approximately half the total time required for the GA/NLP hybrid to converge. Evaluating the CPS best point with tight tolerances yields an objective function of 6.657, which when compared to the best tight tolerance NLP result of 6.891 minutes, shows that the converged results of the two hybrids are of comparable quality. Research is ongoing in improving the reliability of GA global identification for these hybridization studies.

CPS/NLP two-pass hybrid with variable complexity modeling. This study uses the (1.4, 0.5, 0.0) good initial guess for comparison of a CPS/NLP hybrid with CPS and NLP single-algorithm performance from Figure 9. In the hybrid, the CPS initial phase uses inexpensive function evaluations, while the NLP final phase uses tight tolerance, expensive evaluations. Figure 17 shows the optimization history comparison for the CPS/NLP hybrid compared with the benchmark NLP performance. The history jump at the algorithm switch is caused by the change in EPSIT tolerances, which causes an increase in the objective function value at that set of parameter values (from 2.580 at loose tolerances to

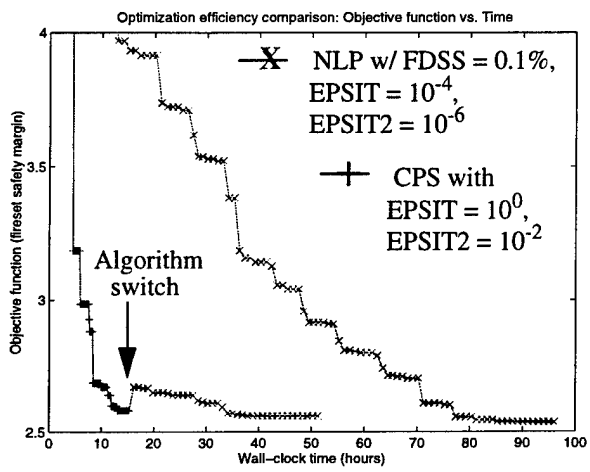


Figure 17. Optimization history comparison of NLP and CPS/NLP hybrid: Best objective function value vs. wall-clock time in hours

2.670 at tight tolerances). Given that the CPS/NLP hybrid achieves an acceptable minimum of 2.560 minutes at tight tolerances, it is observed that wall clock time is reduced by a factor of 2. However, the result achieved is 1% less optimal than the benchmark NLP result of 2.537, which is attributable to the nonsmoothness in the y direction (see Figure 8) in that the hybrid NLP phase gets trapped in the vicinity of y=0.04. The hybrid strategy result is still a considerable improvement over the best CPS result of 2.649 (at tight tolerances). This validates the hybridization strategy in that a more optimal result was computed than was achievable with CPS, and it was achieved in half the time required by NLP.

GA/CPS/NLP three-pass hybrid with variable complexity modeling. Given the results of the GA/CPS, GA/NLP, and CPS/NLP hybridization studies, it appears to be desirable to combine the GA/CPS and CPS/NLP approaches into a three-pass hybrid and address both the global minimum identification problem and the issue of robustness and efficiency to a local minimum. However, the GA global identification performance must first be improved.

Conclusions

Object-oriented software design has been shown to be an effective tool for the generic integration of advanced optimization techniques with broad classes of simulation codes. In a separate paper, applications in nonlinear solid mechanics, heat transfer, fluid mechanics, and structural dynamics were interfaced with existing optimization algorithms via the DAKOTA toolkit[10]. In this paper, fire surety and CVD reactor applications have been employed as benchmarks for demonstration of advanced optimization strategies in algorithm hybridization and parallel processing. These strategies have been designed to be general-purpose and flexible, as enabled by the implementation of generic interfaces in C++. This collection of various algorithms and strategies in the DAKOTA system has allowed for straightforward assessments of relative performance.

In the parallel optimization investigations, significant decreases in wall-clock time have been enabled through the use of parallel computing methodologies. Parallel optimization of single-processor simulations and sequential optimization of massively parallel analyses have been demonstrated in the fire surety and CVD reactor design applications. Peak performance in the fire surety application was prevented by the availability of only 3 commercial QTRAN licenses. In the CVD application, performance was limited by the execution of only one MPSalsa simulation at a time. Since the MPSalsa speed-up tapers off past a certain number of processors, a practical limit is placed on the number of processors per analysis which limits the potential speed-up in this parallel optimization strategy. This points clearly to the need for multiple MPSalsa evaluations running simultaneously in order to achieve peak performance.

In the hybridization investigations, GA/NLP, GA/CPS, and CPS/NLP hybrids have been investigated on the fireset application. In the GA/NLP and GA/CPS hybrids, GA/CPS was shown to be more computationally efficient than GA/NLP in converging to a local minimum, although neither method was successful in navigating to the global minimum due to the difficult global identification problem with the fireset application. More investigation on global identification is needed. Both of these hybrids, however, outperform CPS and NLP single-algorithm performance when these single algorithms are started from an initial guess outside of the global minimum region (Figures 10 and 16). The CPS/NLP hybrid is shown to be an efficient and accurate local convergence technique since a more optimal result was computed than was achievable with CPS alone, and it was achieved in half the time required by NLP alone. Once the global identification problem is better understood, three-pass GA/CPS/NLP hybrids hold promise for combining the performance of the best two-pass approaches.

The overall goal of these research activities is to develop a broadly useful optimization capability with the flexibility and extensibility to easily accommodate broad classes of optimizers, a wide disciplinary range of simulation capabilities, and advanced strategies which seek to enhance robustness and efficiency beyond that which is currently available.

References

[1]Kamat, M.P., Ed., *Structural Optimization: Status and Promise*, Progress in Astronautics and Aeronautics, Vol. 150, American Institute of Aeronautics and Astronautics, Washington DC, 1993.

[2]Törn, A., and Zilinskas, A., *Global Optimization*, Lecture Notes in Computer Science, Springer-Verlag, 1989.

[3]Stroustrup, B., *The C++ Programming Language*, 2nd ed., Addison-Wesley, New York, 1991.

[4]*DOT Users Manual*, Version 4.10, VMA Engineering, Colorado Springs, CO, 1994.

[5]Gill, P.E., Murray, W., Saunders, M.A., and Wright, M.H., *User's Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming*, System Optimization Laboratory, TR SOL-86-2, Stanford University, Stanford, CA, Jan. 1986.

[6]Meza, J.C., "OPT++: An Object-Oriented Class Library for Nonlinear Optimization," Sandia Report SAND94-8225, Sandia National Laboratories, Livermore, CA, March 1994.

[7]Hart, W.E., *Adaptive Global Optimization with Local Search*, Ph.D. dissertation, University of California at San Diego, May 1994.

[8]Hart, W.E., *Evolutionary Pattern Search Algorithms*, Sandia Report SAND95-2293, Sept. 1995.

[9]Haftka, R.T., Gurdal, Z., and Kamat, M.P., *Elements of Structural Optimization*, Second revised edition, Kluwer Academic Publishers, Boston, 1990.

[10]Eldred, M.S., Outka, D.E., Bohnhoff, W.J., Witkowski, W.R., Romero, V.J., Ponslet, E.R., and Chen, K.S., "Optimization of Complex Mechanics Simulations with Object-Oriented Software Design," *Computer Modeling and Simulation in Engineering,* to appear August 1996.

[11]Romero, V.J., Eldred, M.S., Bohnhoff, W.J., and Outka, D.E., "Application of Optimization to the Inverse Problem of Finding the Worst-Case Heating Configuration in a Fire," *Proceedings of the 9th International Conference on Numerical Methods in Thermal Problems*, Atlanta, GA, July 17-21, 1995, Vol. 9, Part 2, pp. 1022-1033.

[12]*P/THERMAL User Manual*, Release 2.6, PDA Engineering, PATRAN Division, Costa Mesa, CA, March 1993.

[13]Ponslet, E.R., and Eldred, M.S., "Discrete Optimization of Isolator Locations for Vibration Isolation Systems," *Proceedings of the 6th Symposium on Multidisciplinary Analysis and Optimization,* paper AIAA-96-4178, Bellevue, WA, Sept. 4-6, 1996.

[14]Jansen, A.N., Orazem, M.E., Fox, B.A., and Jesser, W.A., "Numerical study of the influence of reactor design on MOCVD with a comparison to experimental data," *Journal of Crystal Growth*, 112, 1991, pp. 316-336.

[15]Shadid, J.N., Moffat, H.K., Hutchinson, S.A., Hennigan, G.L., Devine, K.D., and Salinger, A.G., *MPSalsa - A finite element computer program for reacting flow problems. Part 1 - Theoretical background and equation development.* SAND95-2752, Albuquerque, NM (1996).

[16]Salinger, A.G., Devine, K.D., Hennigan, G.L., Moffat, H.K., Hutchinson, S.A., and Shadid, J.N., *MPSalsa - A finite element computer program for reacting flow problems. Part 2 - User's Guide.* In preparation.

[17]Hutchinson, S.A., Shadid, J.N., Moffat, H.K., and Ng, K.T., "A Two-Level Parallel Direct Search Implementation for Arbitrarily Sized Objective Functions," *Proceedings of the Colorado Conference on Iterative Methods*, Breckenridge, Colorado, 1994.

[18]Gangadharan, S.N., Haftka, R.T., and Fiocca, Y.I., "Variable-Complexity-Modeling Structural Optimization Using Response Surface Methodology," paper AIAA95-1164, *36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, New Orleans, LA, April 10-12, 1995.

APPENDIX C

Survey of Approximation Methods in Optimization

Survey of Approximation Methods in Optimization

M. S. Eldred
Sandia National Laboratories

Introduction

When engineering simulations are computationally expensive, optimization must often resort to approximation strategies in order to reduce the number of function evaluations required and also to reduce the computational expense of each function evaluation. The former case of reducing the number of function evaluations involves formal approximation methodologies in which an approximate subproblem is built. The optimizer operates on the subproblem only, and is never directly interfaced with the analysis code. The solution to the approximate subproblem frequently serves as the starting point for a new approximate subproblem, and iteration continues in this way until convergence. This strategy is known as *sequential approximate optimization.*

Whereas the former case encompasses mostly general techniques, the latter case of reducing the computational expense of each function evaluation is highly application dependent and involves tuning a particular analysis to maximize value extraction with respect to computational cost. This technique is separate from the optimization, except for the fact that different optimization algorithms may require different levels of model performance (e.g. response surface smoothness/differentiability).

The following discussions survey the available methodologies in approximation methods for optimization. Recommendations are made for development and a list of references is provided.

Reducing the Number of Function Evaluations

I. Global approximations. This class of approximations is distinguished by the use of many analyses to build approximate representations of response surfaces. Typically, these techniques are most applicable for low dimensional design spaces (a few design variables), and are quite general in that little or no problem-specific knowledge is needed.

   A. Hypersurface fitting. This technique involves fitting a multi-dimensional function (e.g. polynomial, exponential) to data points from exact analyses. Many analyses are generally required. For example, fitting a quadratic polynomial to response data requires at least n(n+1)/2 analyses for n design variables. For expensive analyses, this severely limits the tractable dimensionality of design problems to which this technique can be applied. A related technology is the theory of experiment design, since these methods (e.g. Taguchi, Box) can be used to maximize design space feature extraction with respect to the number of exact analyses by carefully selecting the locations of the numerical "experiments."

   B. Neural networks. This technique involves training an artificial neural network (ANN) with data sets from exact analyses, and then using the trained ANN to return approximate response data for other data sets. Similar to the requirements of surface fitting, accurate training requires many data sets, and the use of experiment design methods for selection of the training sets may increase performance. Accuracy is good for interpolation between

training sets, but can be poor for extrapolation beyond the range of the training sets. In the literature, interpolation capabilities of back-propagation neural networks have been shown to exceed those of surface fitting techniques.

II. Local approximations. This class of approximations involves series representations of objective and constraint functions utilizing function, gradient, and higher-order data from a single point. The techniques are local since their accuracy breaks down away from the vicinity of the analysis point, making it necessary to impose move limits in optimization of the approximate subproblem. Key issues for comparison of local approximations are range of accuracy and required move limits. These techniques are applicable for all design space dimensionalities, but are less general than global approximations, since the most effective local approximation techniques make use of problem-specific knowledge to try to capture the true nonlinear nature of the approximated function.

A. First order local approximations. This class of approximations is based on first-order Taylor series expansions of objective and constraint functions. If the objective function and all constraint functions are linearized and linear programming (LP) optimizers are used to solve the approximate subproblem, the technique is known as *sequential linear programming* (SLP).

1. Linear. This approximation involves a first-order Taylor series expansion of an objective or constraint function in terms of a design variable. This approximation makes no use of problem-specific knowledge and is therefore a general technique. However, depending on the nonlinearity of the approximated function in terms of the design variable, its accuracy can very quickly break down, requiring the use of restrictive move limits.

2. Reciprocal. A Taylor series expansion of an objective or constraint function in terms of the reciprocal of a design variable. The motivation for this approximation is tied to structural optimization of structures consisting of truss or plane-stress elements, since design variables were typically truss element cross-sectional areas (A) or plane-stress element thicknesses (t), and element stresses are much more linear in terms of 1/A or 1/t than in terms of A or t. For general applications, this approximation is useful only when an inverse relation of the approximated function with respect to the design variables is known or suspected.

3. Conservative/Convex. This approximation is a hybrid of the linear and reciprocal approximations and is more conservative than either. This approximation is convex, and when both the objective function and constraints are approximated in this way, the technique is known as *convex linearization*. The resulting approximate subproblem is convex and has a single minimum. The conservative/convex approximation tends to be less accurate than either the direct or reciprocal approximation, so it should only be used when its conservatism or convexity are useful. Methods of moving asymptotes have been introduced to improve accuracy.

4. Intermediate variables & response quantities. These approximations can be very accurate since they make effective use of problem-specific knowledge through the careful selection of intermediate responses and intermediate design variables. The idea

behind this approximation is that of breaking a functional relationship into parts, pulling out any nonlinear explicit portions, and performing the approximation on the most linear implicit relationship available. For the case of stress constraints in structural optimization, stress is a highly nonlinear function of element geometry design variables and a standard linear approximation will require very strict move limits. However, by using element force as a intermediate response (element stress is an explicit function of element force), element section properties as intermediate variables (again, an explicit relationship with the actual design variables), and approximating the remaining relationship between element force and section properties in a linear series, the approximated relationship is very nearly linear (even though the approximated constraint values are highly nonlinear) and generous move limits may be employed. For general application of this technique, the crucial step is the insightful selection of the best intermediate response and variable forms.

B. Higher order local approximations. This technique uses higher-order Taylor series expansions to approximate objective and constraint functions. Quadratic and reciprocal-quadratic approximations are examples that have been investigated in the literature. These approximations are not frequently used, since the improvement in accuracy and the associated reduction in necessary optimization cycles is often not sufficient to offset the increased cost of evaluating higher-order derivatives of response quantities. The well-known technique of *sequential quadratic programming* (SQP) is related in that SQP approximates the Lagrangian function with a second-order Taylor series; however, higher-order derivatives are not actually computed since the second-order term (the Hessian matrix) is approximated by the BFGS update formula.

C. Power law approximations. This approximation is based on selection of an appropriate power relationship of objective and constraint functions in terms of the design variables. This is a general strategy in that it does not require problem-specific knowledge. The assumed form of the relationship is flexible, and appears amenable to the development of an adaptive approximation improvement strategy (start with linear or best guess, and refine the assumed relationship based on the discrepancy between the exact function and its approximation at the end of each approximate subproblem optimization).

D. Differential equation-based approximation. This approach operates on an analytic sensitivity expression by assuming that the coefficients of the various terms of the expression are constant (an approximating assumption in most cases), and then integrating the constant-coefficient differential equation for a high-quality nonlinear approximation. The derived approximations are highly specialized for the particular relationship of interest (no generality) and are most appropriate when the analytic sensitivity expressions are relatively simple.

III. Multipoint approximations. This class of approximations involves series representations of objective and constraint functions which are accurate over larger regions of the design space than most local approximations since they utilize function and gradient data from a few points. Since multipoint approximations are concerned more with matching function and gradient information at a few points and less with capturing nonlinear relationships through problem-specific knowledge, they resemble response surface techniques and can be very

general approaches. In contrast to global response surface approximations, however, the additional analysis points needed to construct the multipoint approximation are few and are usually available through previous analysis points.

Reducing the Expense of a Function Evaluation

I. Model simplification/reduction.

A. Model simplification. This technique entails coarsening mesh densities and removing mesh detail (e.g. cut-outs, threads, or other features requiring a fine mesh) for the purpose of reducing the number of degrees of freedom that must be computed in the simulation. Often these simplified models must be calibrated with respect to a high fidelity model, so that response data of interest in the refined model can be mapped into approximately equivalent data in the simplified model (especially when the response data of interest involve mesh detail which has been removed).

B. Model reduction, condensation, or substructuring. These techniques may be employed to obtain a reduced-order model directly from a refined model without remeshing. For these techniques to be efficient in an optimization context, the use of data describing the refined model should be avoided in the problem formulation. That is, if the reduction employed is not invariant with respect to the design variables in the optimization problem, then the reduction will have to be recomputed after design changes.

C. Variable complexity modeling. This technique is an associated technology when employing models with differing levels of refinement. The concept is that of using the refined model when accurate results are required, and using the inexpensive model when moderate inaccuracy is acceptable. Typically, the simple model is used in a "pre-processor" optimization phase, either being optimized to obtain a good initial design for optimization based on the complex model, being used to "screen out" poor designs which do not merit refined analyses, or being used to define an approximate response surface. Alternatively, the simplified model can be used in a global/local hybrid approximation. In this case, the simplified model can be viewed as a global approximation, since it approximates the true response over the entire design space, and the local approximation occurs when the scale factor defining the ratio between simplified model and refined model responses is approximated with a first-order local approximation. The approximation of the scale factor estimates the variation of the discrepancy between the simplified and refined model responses over the design space. Another potential use which we will investigate combines variable complexity modeling and global/local hybrid optimization algorithms: use of a coarse model in the initial global pre-processor studies (many evaluations, accurate gradients not needed) followed by use of a refined model for local nonlinear programming (fewer evaluations, accurate gradients are required).

II. Adaptive tolerance, time step, and termination time control.

A. Time history extrapolation and adaptive control: Rather than performing a complete transient analysis, the analysis is terminated prematurely, truncating the time histories of response data in order to conserve CPU. The truncated histories are then fit with a function

(polynomial, exponential, etc.) and the function is extrapolated forward in time to compute the desired responses. Of course, the success of this scheme depends highly on the quality of the fit and the smooth behavior of the responses past the truncation point. From experience, this technique is not robust, since the optimizer frequently seeks out those design space regions where the extrapolation breaks down. A much more robust and accurate technique is that of using an adaptive termination time strategy, in which simulation progress is periodically monitored (e.g. by a background script which loops with a time delay) and the simulation is terminated immediately upon capture of the events of interest. This approach is really not an approximation; rather it is an effective strategy for optimizing simulation duration, by both assuring that the simulation proceeds long enough to capture the desired events (avoiding accidental truncation) and by eliminating unnecessary computation for time steps occurring after the event of interest.

B. Convergence tolerance and time step control: In transient simulations, the tightness of convergence tolerances and the size/number of time steps greatly influence the CPU usage of an analysis, the accuracy of the results, and the small-scale smoothness of the response variations in the design space. Transient analyses with large time steps and loose tolerances are inexpensive, but suffer from inaccuracy and small-scale nonsmoothness. Likewise, analyses with small time steps and tight tolerances are accurate and smooth, but can be prohibitively expensive in CPU usage. Thus, an important concern in reducing the expense of function evaluations is the trade-off between model performance (accuracy, smoothness of response variations) and CPU usage. Strategies for achieving an effective balance include (1) perform trade-off study off-line to determine best control settings, (2) adaptively control settings based on smoothness metrics during the optimization process, and (3) variable-complexity modeling: employ large time steps and loose tolerances for initial studies and tighten controls for refined studies.

Recommendations

One of the key concerns for incorporating approximation methodologies within the DAKOTA software is the generality of the technique. In addition to having good performance, the developed approximation methods should be broadly applicable in different engineering disciplines without the need for application-specific modifications. Many of the cited approximation methods grew out of the structural optimization research community, and as such, they take advantage of structures-specific knowledge in order to best capture nonlinear relations in local approximations. The DAKOTA toolkit has a breadth of application that is much wider than structures, which necessarily makes the incorporation of problem-specific knowledge into approximation methods a difficult proposition.

For reducing the number of function evaluations, global approximation techniques look promising as *general* approximation strategies for those problems when the number of design variables is approximately 10 or less. For these cases, DAKOTA should be equipped with response surface or neural network approximation capabilities, and this is recommended as a development item. Additionally, it would be desirable to incorporate methods of experiment design as pre-processors in response surface generation and neural network training, so as to maximize design space feature extraction for a limited number of computational "experiments." The most effective local approximation techniques are not general; they require problem-specific

knowledge for determining the best approximated response and series parameter forms. Thus, only the simplest of local approximation strategies (the linear and power law approximations) could be incorporated into DAKOTA as *general* approximation techniques. The development of sophisticated means for incorporating problem-specific knowledge into a intermediate variable/ response approximation within DAKOTA has high potential payoff, but the relative difficulty and high risk of the task makes it difficult to assess any priority to its development. More realistic targets for development within DAKOTA are the techniques of multipoint approximations, which can be highly accurate, efficient, and general. Therefore, development of response surfaces and neural networks with experiment design pre-processing, linear and power law local approximations, and multipoint approximations have been identified as the best candidates for development and inclusion in the DAKOTA toolkit.

In terms of reducing the expense of a given function evaluation, the cited techniques of model simplification/reduction and adaptive tolerance, time step, and termination time control have been shown to be very effective in application work to date and are recommended for continued use and development. Since they are highly application-specific, incorporation into the DAKOTA software is not generally feasible, and they will continue to be effected in the filter and analysis driver programs (adaptive strategies) or off-line from the DAKOTA optimization executions (nonadaptive strategies).

References

Global and local approximations:

Barthelemy, J.-F.M., and Haftka, R.T., "Function Approximations," *Structural Optimization: Status and Promise*, Kamat, M.P., ed., American Institute of Aeronautics and Astronautics, Washington, DC, 1993, pp. 51-70.

Haftka, R.T., Gurdal, Z., and Kamat, M.P., *Elements of Structural Optimization*, Kluwer Academic Publishers, Boston, MA, 1990, pp. 177-209.

Kirsch, U., "Improved Stiffness-Based First-Order Approximations for Structural Optimization," *AIAA Journal*, Vol. 33, No. 1, Jan. 1995, pp. 143-150.

Vanderplaats, G.N., and Salajegheh, E., "New Approximation Method for Stress Constraints in Structural Synthesis," *AIAA Journal*, Vol. 27, No. 3, March 1989, pp. 352-358.

Sepulveda, A.E., and Thomas, H.L., "New Approximation for Steady-State Response of General Damped Systems," *AIAA Journal*, Vol. 33, No. 6, June 1995, pp. 1127-1133.

Prasad, B., "Novel Concepts for Constraint Treatments and Approximations in Efficient Structural Synthesis," *AIAA Journal*, Vol. 22, No. 7, July 1984, pp. 957-966.

Multipoint approximations:

Fadel, G.M., Riley, M.F., and Barthelemy, J.-F.M., "Two Point Exponential Approximation Method for Structural Optimization," *Structural Optimization*, Vol. 2, Nos. 1-2, 1990, pp. 117-124.

Rasmussen, J., "Accumulated Approximations - A New Method for Structural Optimization by

Iterative Improvements," *3rd Air Force/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*, San Francisco, CA, Sept. 1990, pp. 253-258.

Polynkin, A.A., van Keulen, F., and Toropov, V.V., "Optimization of geometrically nonlinear thin-walled structures using the multipoint approximation method," *Structural Optimization*, April 1995, pp. 105-116.

Experiment design methods (Box, Taguchi):

Box, G.E.P., and Draper, N.R., *Empirical Model-Building and Response Surfaces*, Wiley, New York, 1987.

Yurkovich, R., "The Use of Taguchi Techniques with the ASTROS code for Optimum Wing Structural Design," paper AIAA-94-1484-CP, *Proceedings of the 35th AIAA/ASME/ASCE/ AHS/ASC Structures, Structural Dynamics and Materials Conference*, Hilton Head, SC, April 18-20, 1994, pp. 1334-1342.

Ross, P.J., *Taguchi Techniques for Quality Engineering*, McGraw-Hill, New York, 1988.

Neural networks for approximate analysis:

Hajela, P., and Berke, L., "Neural Networks in Structural Analysis and Design: An Overview," *Computing Systems in Engineering*, Vol. 3, Nos. 1-4, 1992, pp. 525-538.

Szewczyk, Z.P., and Hajela, P., "Function Extrapolation in Structural Reanalysis Using Neural Networks," paper AIAA-94-1599-CP, *Proceedings of the 35th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Hilton Head, SC, April 18-20, 1994, pp. 2247-2256.

Variable complexity modeling:

Gangadharan, S.N., Haftka, R.T., and Fiocca, Y.I., "Variable-Complexity-Modeling Structural Optimization Using Response Surface Methodology," paper AIAA95-1164, *36th AIAA/ ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, New Orleans, LA, April 10-12, 1995.

Gage, P.J., Kroo, I.M., and Sobieski, I.P., "A Variable-Complexity Genetic Algorithm for Topological Design," paper AIAA94-4413, *5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, Sept. 7-9, 1994.

Hutchison, M.G., Unger, E.R., Mason, W.H., Grossman, B., and Haftka, R.T., "Variable-Complexity Aerodynamic Optimization of a High Speed Civil Transport Wing," *Journal of Aircraft*, Vol. 31, No. 1, Jan.-Feb. 1994, pp. 110-116.

Model reduction, condensation, and substructuring:

Noor, A.K., "Recent advances and applications of reduction methods," *Applied Mechanics Reviews*, vol. 47, no. 5, May 1994, pp. 125-146.

Meirovitch, L., *Computational methods in structural dynamics*, Sijthoff & Noordhoff, Rockville, Maryland, 1980.

APPENDIX D

The Discrete Adjoint Method for Gradient Evaluation in Optimization

The Discrete Adjoint Method for Gradient Evaluation in Optimization

C. D. Moen
Computational Reacting Processes Department
Sandia National Laboratories, California
October 1997

Abstract

The discrete adjoint method provides a less expensive means for generating gradient information in optimization. A large expensive analysis code is run once for each design variable when evaluating numerical derivatives of the objective function and nonlinear constraints. With the adjoint method, the analysis code needs only to be run once per gradient evaluation. The additional adjoint information is extracted from the solution procedure existing in the analysis code. When there are nonlinear constraint equations, the adjoint method is beneficial only if the number of design variables is much larger than the number of constraints. A derivation of the adjoint method is given as applied to optimal design problems. Practical implementation issues are discussed.

Table of Contents

Nomenclature

Roman symbols

n_d	=	number of design variables, scalar
n_c	=	number of nonlinear constraints, scalar
n_l	=	number of linear iterations of iterative matrix solver, scalar
n_n	=	number of nonlinear iterations of iterative solution procedure, scalar
$\mathbf{x}$	=	design variables, vector
$\mathbf{x}_l$	=	lower bounds on design variables, vector
$\mathbf{x}_u$	=	upper bounds on design variables, vector
$\mathbf{q}$	=	solution variables, vector
$\tilde{\mathbf{q}}_i$	=	solution variables for perturbed design variable x_i, vector
F	=	objective function, scalar
$\mathbf{C}$	=	general nonlinear constraints, vector
$\mathbf{G}$	=	gradient of the objective function, vector
$\mathbf{H}$	=	Hessian of the objective function, matrix
$\mathbf{R}$	=	residual of the physical conservation laws, solved by the analysis code, physical constraints, vector
$\mathcal{L}$	=	Lagrangian operator, vector
$\mathbf{A}$	=	stiffness matrix or Jacobian, matrix
$\mathbf{b}$	=	load vector, vector
$\mathbf{p}$	=	Krylov vector

Greek symbols

λ	=	Lagrange multipliers or co-state variables, tensor
ϵ_i	=	perturbation to ith component of $\mathbf{x}$, scalar

Superscripts

T	=	transpose

Subscripts

d	=	pertaining to design variables
c	=	pertaining to constraint equations
l	=	pertaining to linear iterations
n	=	pertaining to nonlinear iterations

Chapter 1: Introduction

This report represents a portion of worked performed under a joint three-year LDRD on optimization. The purpose of this project is to investigate methods for computing gradient information more cheaply.

In the past, we have studied optimal design problems for thermal processing equipment used in the microelectronics manufacturing industry. We used gradient-based methods to determine optimal operating parameters to maximize process uniformity [1, 2]. Gradient information was generated by applying divided differences to perturbed solutions from large, expensive finite-element analysis codes. Each function evaluation for the optimization is expensive since the analysis codes can take upwards of an hour of computer time. Less expensive methods of gradient evaluation are desired.

We investigated automatic differentiation for generating analytic gradients directly in the analysis code [3]. We found that there is a minimum problem size below which the overhead of automatic differentiation makes it more expensive than divided differences. Additionally, the window for usefulness is small since for larger optimization problems, above the efficiency threshold, one rapidly runs out of computational resources.

The adjoint method reduces the amount of work required to evaluate gradients. The adjoint method, based on variational principles, has proven useful in shape optimization for aerodynamic design [4, 5, 6, 7]. There have been both continuous and discrete approaches to forming the adjoint equations. While the continuous approach provides an equation set for the adjoint problem, the discrete meshing sensitivity is lost and boundary conditions for general objective functions are problematic. We investigate the discrete adjoint formulation because it does incorporate the mesh dependence from the analysis code, and the objective function and constraints are easy to implement. Also, the discrete adjoint method is implemented in such a way that minimal modifications are required for validated, production analysis codes.

The advantage of the adjoint method is that gradients are constructed from the sensitivity information that is inherent in an analysis code. The sensitivity information is ignored in brute-force numerical differencing methods. Most analysis codes solve nonlinear equations by iterating on a linearization of the governing equations. The information contained in the

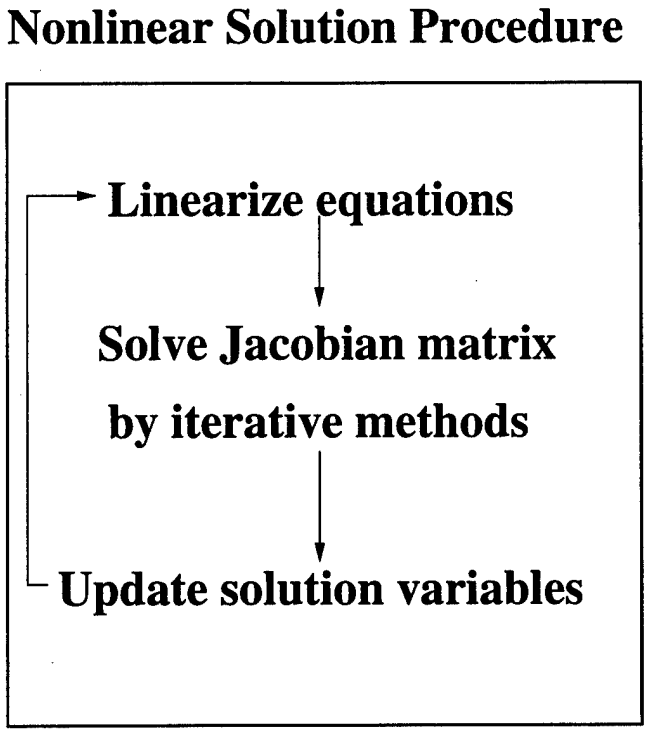


Figure 1.1 Solution Procedure Paradigm Required for Efficient Application of Adjoint Method

linear problem is the sensitivity information that the adjoint method requires.

The adjoint method works best with analysis codes that have a solution procedure based on Newton or Picard iteration: outer nonlinear iterations over an inner linearized problem. The paradigm is shown in Figure 1.1. It is implied that all equations are coupled together in the linearization which is typically the case for finite element codes, such as COYOTE [8, 9] and SALSA [10, 11]. It is expensive to store all the matrix coefficients from the linearization when the mesh becomes large or the number of physical variables is large (reacting flow, for an example). It is more common to find solution procedures which approximate the linearization by breaking the matrix into pieces.

The segregated linearization approach is used when there are several different governing equations. The equations are solved sequentially. A principal solution variable is associated with each physical equation and the linearization of the equation if performed with respect to the variable. The segregated approach is found most often in fluid mechanics code, such as CURRENT [12], where velocities are associated with momentum equations and temperature is associated with an energy equation. The individual linearizations can be combined to form the overall sensitivities required for the adjoint method, but the sensitivity coefficients are approximate so the gradients will be approximate. An outstanding question is whether the

approximate gradients are accurate enough to be useful for the optimization. Sometimes the segregated solution approach is applied in an embedded manner. The solution of radiative fluxes in the TACO code [13] is segregated and embedded in the solution for the temperatures.

Another possibility for approximating the linear problem is matrix factorization. Many fluid mechanics codes are constructed with such a solution strategy. Some people researching adjoint methods have steered away from using approximate factorization because it is too difficult to form the transpose.

In the following investigation of the adjoint method, the approach will to use any sensitivity information that is available from an analysis code. The effectiveness of using approximate sensitivity information is assessed.

Chapter 2: Gradient Evaluation

Our optimization problems are described in terms of an objective function, subject to bounds on the design variables and general nonlinear constraints. The design variables, $\mathbf{x}$, are the values we would like to optimize such that some objective function, F, is minimized. The objective function is a function of both the design variables, $\mathbf{x}$, and other solution variables, $\mathbf{q}$, determined by the analysis code. The analysis code introduces additional physical constraints on allowable combinations of the design variables and solution variables through the residual of the conservation laws, $\mathbf{R}$. The mathematical description of the optimization problem is

$$\min_{\mathbf{x}} \; F(\mathbf{x}, \mathbf{q}(\mathbf{x})) \tag{2.1}$$

subject to the nonlinear constraints

$$\mathbf{C}(\mathbf{x}, \mathbf{q}) < 0, \tag{2.2}$$

the lower and upper bounds on design variables

$$\mathbf{x}_l < \mathbf{x} < \mathbf{x}_u, \tag{2.3}$$

and the physical analysis constraints on the solution variables

$$\mathbf{R}(\mathbf{x}, \mathbf{q}) = 0. \tag{2.4}$$

The general nonlinear constraints $\mathbf{C}$ are separated from the physical constraints $\mathbf{R}$ because constraints associated with the optimization objective are enforced differently than constraints associated with the analysis code.

The gradients of the objective function and general nonlinear constraints are required for gradient-based optimization methods. Generally, the objective function and constraints are written in terms of the geometric design variables and the solution variables. The solution variables are also functions of the design variables. The gradients of the objective function and constraints with respect to the design variables are calculated using the following analysis. If the design variables and solution variables were independent, then

the differentials of the objective function and nonlinear constraints at an arbitrary point $(\mathbf{x}, \mathbf{q})$ would be

$$dF = \frac{\partial F}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial F}{\partial \mathbf{q}} d\mathbf{q}, \tag{2.5}$$

$$d\mathbf{C} = \frac{\partial \mathbf{C}}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial \mathbf{C}}{\partial \mathbf{q}} d\mathbf{q}, \tag{2.6}$$

but the directional derivatives are meaningless unless one knows how the solution variables change with the design variables. When the physical constraints are satisfied, $\mathbf{R}(\mathbf{x}, \mathbf{q}) = 0$, then there is a dependence between $\mathbf{x}$ and $\mathbf{q}$ such that $\mathbf{q} = \mathbf{q}(\mathbf{x})$. The gradients can be written as

$$\nabla \mathbf{F} = \frac{\partial F}{\partial \mathbf{x}} + \frac{\partial F}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \mathbf{x}}, \tag{2.7}$$

$$\nabla \mathbf{C} = \frac{\partial \mathbf{C}}{\partial \mathbf{x}} + \frac{\partial \mathbf{C}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \mathbf{x}}. \tag{2.8}$$

Equations 2.7 and 2.8 are only valid when the solution variables satisfy the physical constraints, Equation 2.4.

The expensive part of evaluating the gradient is computing the sensitivity derivatives in Equations 2.7 and 2.8. Three methods for evaluating the gradient are presented in the following sections—the "black-box" method, the adjoint method, and the direct method. Shubin and Frank [14] showed that the discrete forms of each of these methods are mathematically equivalent, though the implementation expense is difference.

2.1 Black-Box Method for Numerical Gradients

Previously, we have used a "black-box" approach to solving the optimization problem defined by Equations 2.1 through 2.4. Solutions to the problem $\mathbf{R}(\mathbf{x}, \mathbf{q}) = 0$ were used to generate objective function values. Gradients of the objective function are computed through divided differences of the objective function. When there are n_d design variables defining $\mathbf{x}$, then $n_d + 1$ expensive nonlinear problems, $\mathbf{R}$, must be solved to generate the gradient of F with respect to $\mathbf{x}$. Sequentially, each design variable x_i is perturbed by ϵ_i. The solution vector, $\tilde{\mathbf{q}}_i$, resulting from the solution of the perturbed system of equations, is used to find the derivative of the objective function.

$$\nabla F \approx \frac{F\left(\mathbf{x} + \epsilon_i, \tilde{\mathbf{q}}_i; \mathbf{R}\left(\mathbf{x} + \epsilon_i, \tilde{\mathbf{q}}_i\right) = 0\right) - F\left(\mathbf{x}, \mathbf{q}; \mathbf{R}\left(\mathbf{x}, \mathbf{q}_i\right) = 0\right)}{\epsilon_i} \tag{2.9}$$

The black-box procedure implicitly contains the sensitivity of the solution variables to the design variables, $\partial \mathbf{q} / \partial \mathbf{x}$.

2.2 Adjoint Method for Numerical Gradients

The adjoint method is a variational method that is used to extract additional information from the analysis problem to computationally simplify the construction of gradients. The method is grounded in the formalism of optimal control theory [4] and an extended history of the method, as applied to optimization and computational fluid dynamics, is given by Anderson and Venkatakrishnan [7]. The minimization of the objective function is reformulated as a Lagrange multiplier problem, explicitly including the physical constraint relations, $\mathbf{R}$. The Lagrangian is

$$\mathcal{L} = \begin{bmatrix} F(\mathbf{x}, \mathbf{q}) \\ C(\mathbf{x}, \mathbf{q}) \end{bmatrix} - \lambda^T \mathbf{R}(\mathbf{x}, \mathbf{q}), \tag{2.10}$$

where λ is the Lagrange multiplier. Note that in this formulation, the $\mathbf{x}$ and $\mathbf{q}$ values are considered to be independent and the minimization problem is defined by

$$\min_{\mathbf{x},\mathbf{q},\lambda} \mathcal{L}(\mathbf{x}, \mathbf{q}, \lambda). \tag{2.11}$$

The conditions for the adjoint formulation are defined by:

$$\frac{\partial \mathcal{L}}{\partial \lambda} = -\mathbf{R} = 0 \tag{2.12}$$

$$\frac{\partial \mathcal{L}^T}{\partial \mathbf{q}} = \frac{\partial}{\partial \mathbf{q}} \begin{bmatrix} F \\ \mathbf{C} \end{bmatrix}^T - \frac{\partial \mathbf{R}^T}{\partial \mathbf{q}} \lambda = 0 \tag{2.13}$$

$$\frac{\partial \mathcal{L}^T}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \begin{bmatrix} F \\ \mathbf{C} \end{bmatrix}^T - \frac{\partial \mathbf{R}^T}{\partial \mathbf{x}} \lambda \tag{2.14}$$

The first equation states that the design variables $\mathbf{x}$ and solution variables $\mathbf{q}$ must be consistent. The second equation is the adjoint equation by definition. The third condition defines the gradient and will be equal to zero if the gradient is equal to zero.

In the adjoint method solution procedure, Equation 2.12 is first solved for $\mathbf{q}$ given a set of design variables $\mathbf{x}$. The adjoint problem, Equation 2.13, is then solved for the co-state variables λ. If λ also satisfies Equation 2.14, then the design variables are optimal. Otherwise, the derivative of the Lagrangian in Equation 2.14 is equivalent to the gradient of the objective function, Equation 2.7. (Note: in Equation 2.14, the derivative $\partial F/\partial \mathbf{x}$ is for a fixed value of $\mathbf{q}$.)

The adjoint method requires the evaluation of four Jacobian matrices: $\partial F/\partial \mathbf{x}$, $\partial F/\partial \mathbf{q}$, $\partial \mathbf{R}/\partial \mathbf{x}$, and $\partial \mathbf{R}/\partial \mathbf{q}$. The conservation law Jacobian, $\partial \mathbf{R}/\partial \mathbf{q}$, already exists in some form as part of the solution procedure in the analysis code. The derivatives of the residual with respect to the design variables are computed through divided differences of the residual. The

conservation-law residual is available from the analysis code and is inexpensive to compute. The derivatives of the objective function are either derived analytically or computed with divided differences.

The adjoint method is less expensive than the black-box method because it only requires residual evaluations and linear matrix solves. The black-box method requires full nonlinear iterations. If there many more nonlinear constraints than design variables, then the adjoint method becomes expensive. Each constraint equation has a set of co-state variables that are computed by inverting the linear adjoint equations.

2.3 Direct Method for Numerical Gradients

Recall that the gradient information can be computed from Equations 2.7 and 2.8 if the sensitivity derivatives, $\partial\mathbf{q}/\partial\mathbf{x}$, are known. Consider the differential of the conservation law residual,

$$\mathrm{d}\mathbf{R} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}}\mathrm{d}\mathbf{x} + \frac{\partial \mathbf{R}}{\partial \mathbf{q}}\mathrm{d}\mathbf{q}. \tag{2.15}$$

For solutions of Equation 2.4, the differentials $\mathrm{d}\mathbf{R}$ are zero and

$$\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \mathbf{x}} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}}. \tag{2.16}$$

This is a linear system requiring matrix solves for the n_d vector components of $\partial\mathbf{q}/\partial\mathbf{x}$. The work required to generate the Jacobian matrices is the same between the direct and adjoint methods. The difference between the two methods is the number of linear matrix inversions.

2.4 Direct Method is Mathematically Equivalent to Adjoint Method

In this section, proof is given that the derivative of the Lagrangian with respect to the design variables is equivalent to the gradient of the objective function and nonlinear constraints. Direct differentiation is first applied to the Lagrangian,

$$\mathrm{d}\mathcal{L} = \left(\frac{\partial}{\partial \mathbf{x}}\begin{bmatrix} F \\ \mathbf{C} \end{bmatrix} - \lambda\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\right)\mathrm{d}\mathbf{x} + \left(\frac{\partial}{\partial \mathbf{q}}\begin{bmatrix} F \\ \mathbf{C} \end{bmatrix} - \lambda\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right)\mathrm{d}\mathbf{q}. \tag{2.17}$$

By definition, we solve the adjoint problem Equation 2.13. Removing the adjoint equation from Equation 2.17 leaves only the differential dependence on the design variables,

$$\mathrm{d}\mathcal{L} = \left(\frac{\partial F}{\partial \mathbf{x}} - \lambda\frac{\partial \mathbf{R}^T}{\partial \mathbf{x}}\right)\mathrm{d}\mathbf{x}, \tag{2.18}$$

or, substituting for the solution of λ,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}}\begin{bmatrix} F \\ \mathbf{C} \end{bmatrix} - \frac{\partial}{\partial \mathbf{q}}\begin{bmatrix} F \\ \mathbf{C} \end{bmatrix}\frac{\partial \mathbf{R}^{-1}}{\partial \mathbf{q}}\frac{\partial \mathbf{R}}{\partial \mathbf{x}}. \tag{2.19}$$

Substituting Equation 2.16 into Equation 2.19 gives the final form of the gradient of the Lagrangian

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \begin{bmatrix} F \\ \mathbf{C} \end{bmatrix} - \frac{\partial}{\partial \mathbf{q}} \begin{bmatrix} F \\ \mathbf{C} \end{bmatrix} \frac{\partial \mathbf{q}}{\partial \mathbf{x}}, \tag{2.20}$$

which is equivalent to Equations 2.7 and 2.8.

Chapter 3: Adjoint Method Implementation

The computational expense of the adjoint method varies relative to the "black-box" method depending on the number of design variables and constraints, and the solution procedure used in the analysis code. In implementing the discrete adjoint method, we wish to make use of as much existing information from the analysis code as possible. Also, we want to access the information in such a way that we minimize the amount of computer code we modify or add. First, a definition of the linear problem is required. The residual of the discrete governing equations is $\mathbf{R}$. The residual is usually linearized about the solution variables, $\mathbf{q}$, to form the linear problem $\mathbf{Aq} = \mathbf{b}$. Note that if Newton linearization is used, then $\mathbf{b} = \mathbf{R}$ evaluated at the previous solution and $\mathbf{q}$ is the change is solution variables. In the following derivations, $\mathbf{q}$ are assumed to be the solution variables.

Consider the computational work required to form the gradient of the objective function using direct differentiation with numerical forward differences. For n_d design variables, the analysis code must be run $n_d + 1$ times. Assume the analysis code uses an implicit solution technique consisting of repeated steps of evaluating the residuals of the conservation laws and building a matrix of implicit terms. Each run of the analysis code consists of n_n outer iterations to solve a nonlinear problem. Each nonlinear iteration requires a matrix construction step, T_k, and n_l inner iterations of a linear solver T_a. (For this work estimate, it is assumed that the entire matrix is stored. Some solution procedures continuously reconstruct parts of the matrix during the solution procedure in which case the cost is proportional to $n_l\,(T_k + T_a)$.) The linear solve usually consists of iterative operations similar to matrix-vector multiplies. If T_k and T_a are the work-loads associated with the matrix-build and one inner iteration of the linear solve, then the overall work required to form a gradient is proportional to $T_D = (n_d + 1)\,n_n\,(T_k + n_l T_a)$.

To form a gradient with the adjoint method, the analysis code must first be run for the current state of design variables to extract the solution variables. The adjoint problem then consists of forming the matrix and inverting once for each objective function and nonlinear constraint, n_c. The discrete adjoint matrix is the transpose of the Jacobian matrix, $\partial\mathbf{R}/\partial\mathbf{q}$. The Jacobian matrix is often available as the matrix of implicit coefficients from the analysis code. The derivatives of the residual with respect to the design variables, $\partial\mathbf{R}/\partial\mathbf{x}$ are

computed by divided differences. The procedure is fast since only the residuals are evaluated and no matrix inversions are required. The overall work required to form a gradient is proportional to $T_A = n_n\left(T_k + n_l T_a\right) + \left(T_k + n_l T_a\right) + \left(T_k + n_c n_l T_a\right) + \left(n_d + 1\right)\left(T_k + T_a\right)$.

For most analysis codes, the time to form the matrix, T_k, is small relative to the time required to invert the matrix, $n_l T_a$. In the limit that $T_k \ll n_l T_a$, the ratio of times is

$$\frac{T_A}{T_D} \approx \text{constant} + \frac{1 + n_c/n_n}{1 + n_d}. \tag{3.1}$$

If the number of design variables is larger than the number of nonlinear constraints, then the adjoint method is the less expensive method. If the number of design variables is much smaller than the number of nonlinear constraints, the direct differentiation method is less expensive. Note that the dependence on the number of constraints is scaled by the expense of performing a nonlinear analysis. There can be a large number of constraints and the adjoint method will still be beneficial if the analysis code requires a lot of iteration to extract a solution. It is assumed that the adjoint matrix is relatively easy to form and invert so that the expense of the transpose step does not factor into the timing analysis.

In the following sections, different linearization schemes are discussed in relation to the adjoint method.

3.1 Using Full-Matrix Analysis Codes

The easiest implementation of the discrete adjoint method is for the case where the full Jacobian matrix is available from the analysis code. An example of such a code that we use is the COYOTE [8, 9] finite-element heat transfer code. The stiffness matrix for the heat conduction problem, including radiation flux terms at boundaries, is stored as a sparse matrix. The code has been modified to solve the adjoint problem by transposing the stiffness matrix in sparse form and using the built-in preconditioned gradient schemes to solve Equation 2.13. The derivatives $\partial\mathbf{R}/\partial\mathbf{x}$ are computed by divided differences. The residual, $\mathbf{R}$, is computed by $\mathbf{R} = \mathbf{A}\mathbf{q} - \mathbf{b}$, where $\mathbf{A}$ is the stiffness matrix and $\mathbf{b}$ is the load vector. A mesh is constructed for each perturbed set of mesh parameters using the FASTQ [15] mesh generator, run in batch mode.

3.2 Using Analysis Codes with Embedded Solvers

When enclosure radiation problems are solved in heat transfer, the solution of the radiosities is usually split from the solution of the heat conduction. First, the radiation heat flux problem between surfaces is solved for a fixed temperature field. The heat fluxes are then

added to the load vector for the conduction problem. This procedure can be viewed as a matrix inversion embedded within the right-hand-side of the conduction matrix. Two such heat transfer codes are the TACO code [13] and the COYOTE [8, 9] code (COYOTE has the option of either treating the enclosure radiation problem as embedded or fully-coupled).

It is probable that transposing only the conduction matrix for the adjoint method will not provide enough information to generate an accurate gradient. Somehow, the surface exchange information contained in the load vector must be introduced, inexpensively.

3.3 Using Analysis Codes with Segregated Solvers

The CURRENT [12] code uses a segregated solution approach. The continuity, momentum, and energy equations are solved solved at separate steps for the solution variables of pressure, velocity, and temperature. If the objective function is only written in terms of a single solution variable (an objective function based on prescribed heat flux only depends explicitly on the temperature variable), then it may be possible to generate gradient information from the linearization of a single equation (the energy equation in this case).

The matrix coefficients for an equation set in CURRENT are stored in terms of stencil positions for each control volume. To form the transpose coefficient set, the north coefficient of a control volume maps to the south coefficient of the north adjacent control volume, the south coefficient maps to the north coefficient of the south adjacent control volume and so on. Boundary conditions are folded in the source term so only internal control volumes are involved. The line-relaxation scheme can then be applied to the transposed coefficient set. Care must be taken to remove the time-damping terms from the coefficient sets before applying the transpose operation.

3.4 Using Analysis Codes with Matrix-Free Solvers

If the transpose of the linear matrix operator is difficult to generate, a matrix-free method may be considered. The matrix-free approach will not be cost-effective unless the overhead of starting and initializing (command parsing and data structure evaluation) an analysis code is low. A matrix-free gradient method can be used to invert the adjoint problem, Equation 2.13. The Krylov vectors for the matrix-free gradient method are generated using numerical differentiation.

First, consider a matrix-free gradient method for solving $\mathbf{Aq} = \mathbf{b}$, where $\mathbf{A} = \partial\mathbf{R}/\partial\mathbf{q}$. Krylov vectors for the gradient method are generated by multiplying the matrix $\mathbf{A}$ by

previous search direction vectors, $\mathbf{p}$. Numerically,

$$\mathbf{Ap} = \frac{\partial \mathbf{R}_i}{\partial \mathbf{q}_j}\mathbf{p}_j \tag{3.2}$$

$$= \left(\mathbf{R}_i(\mathbf{q}_j + \epsilon \mathbf{p}_j) - \mathbf{R}_i(\mathbf{q}_j)\right)/\epsilon \tag{3.3}$$

The matrix-vector product is generated by differencing residual functions based on the solution vector and the solution vector perturbed by the search direction. The operation is simple because solution vector is perturbed and then the residual is evaluated with the perturbed solution vector. When the transpose matrix is involved, the perturbation process becomes more difficult.

$$\mathbf{A}^T\mathbf{p} = \frac{\partial \mathbf{R}_j}{\partial \mathbf{q}_i}\mathbf{p}_j \tag{3.4}$$

$$= \left(\mathbf{R}_j(\mathbf{q}_i + \epsilon) - \mathbf{R}_j(\mathbf{q}_i)\right)\mathbf{p}_j/\epsilon \tag{3.5}$$

The analysis code must be restarted with a different perturbation for each equation which can be expensive, or the perturbation must be hardwired into the code at the equation evaluation level which is usually a long-and-involved process.

The matrix-free was not tried because all the analysis codes of interest have high start-up overhead.

Chapter 4: Applications

The adjoint method is tested with two constrained shape optimization applications with heat transfer design. The first application is a shape design of a high-conductivity material insert for cooling packages. The COYOTE code is used for the analysis and the transpose matrix is directly available from the code. The second application is a flow channel design to optimize the convective heat flux to a channel wall. The CURRENT code is used for the analysis and the transpose matrix information is extracted from the energy equation.

For all optimizations, the DAKOTA [16] package is used to drive the optimization process. The gradients from the adjoint method are computed external to DAKOTA and supplied as "analytic gradients".

4.1 Cooling Package Design

The operational power of RF amplifiers, used for cellular telephone base-stations, is limited by the temperature of the active area. Each amplifier package is mounted on a cold block, but the top-side temperature of the active device is limited by thermal conduction through the package to the cooler base. To run at higher RF powers, a greater amount of cooling must be applied. It would be expensive to redesign or retro-fit the base-station cooling. Instead, the individual RF packages can be redesigned to improve the conductive cooling path to the base-station cooling mechanism. A solution to the conduction problem is to add an insert of high-conductivity material below the active area in the RF package. Commercially manufactured diamond has been considered because of high thermal conductivities near 10 W/cm/K. Unfortunately, diamond material is also very expensive so we must design a cooling insert with the minimum amount of diamond to achieve a desired device temperature.

4.1.1 Optimization Problem

The optimization problem is to minimize the volume of the diamond insert given a nonlinear constraint on the temperature of the active device. There are three design variables describing the size and position of the insert. There are lower and upper bounds on the design variables, constraining the insert design to fit within the RF package. The temperature for

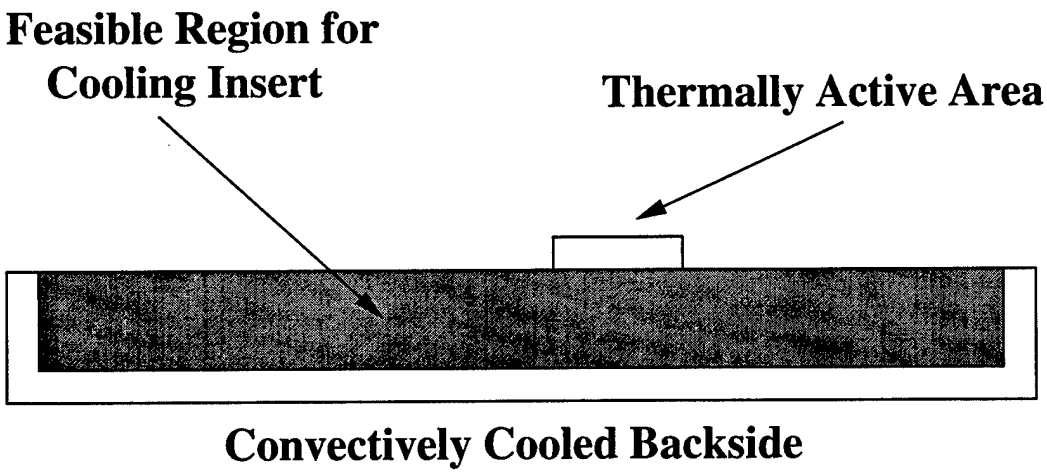


Figure 4.1 Diamond Insert Provides Improved Conduction Path from Hot Active Area to Cooled Base Plate

the nonlinear constraint is computed with the COYOTE [8, 9] finite-element heat transfer code and the FASTQ [15] meshing code.

A schematic of the RF package is given in Figure 4.1. The bottom rectangular shape is the package material, a copper-tungsten alloy. The package is 385 mil wide and 60 mil deep. The active material is the small rectangle of silicon, mounted on top of the copper-tungsten. The silicon die is 35 mil wide and 5 mil deep. The gray area in the copper-tungsten block is the feasible region in which to place the diamond insert, also of rectangular shape. The size of the diamond is constrained such that it is at least 5 mil deep and it must be at least 5 mil from the sides and bottom of the copper-tungsten package. Also, the diamond insert must at least cover the area directly beneath the active area.

For a given RF operational power, there will be a certain amount of heat dissipated within the device. The heat generation is introduced as a heat flux across the top of the silicon die.

4.1.2 Design Analysis

The maximum allowable die temperature is 150°C. The bottom of the RF package is cooled with a convective cooling coefficient of 7.75 $W/cm^2/K$ and a far-field temperature of 50°C. The power dissipation at the die surface without the diamond insert is 1830 W/cm^2. It is desired to increase the RF power such that the dissipated power is 2590 W/cm^2. The resulting die temperature would be an average of 192°C, which is too high.

The cooling package was meshed with block-structured grids. The block boundaries move with the material interfaces. The number of grid points and the relative grid spacing is fixed

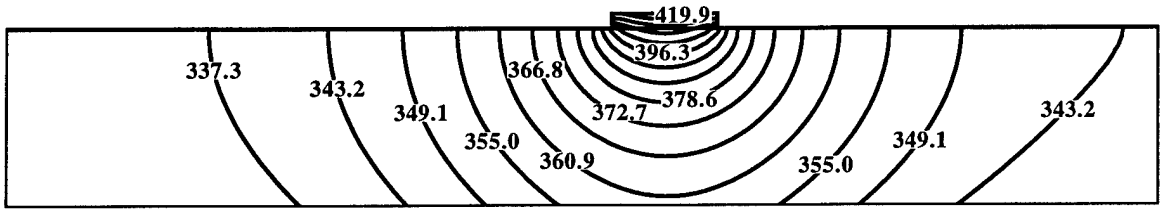


Figure 4.2 Temperature (K) Contours for Baseline RF Package at 1830 W/cm^2

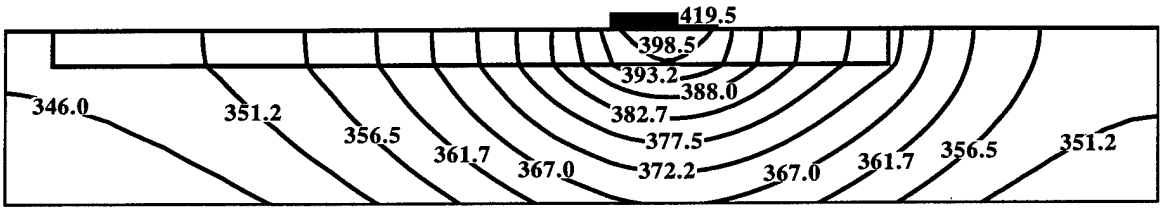


Figure 4.3 Temperature (K) Contours for RF Package at 2950 W/cm^2 with Diamond Insert

in each block.

The temperature contours through the RF package for the baseline design are shown in Figure 4.2. The optimal placement of the diamond insert is as a sheet extending to the open area at the left of the die, shown in Figure 4.3. The diamond sheet spreads the heat out so is sees more cooled surface area. This is an interesting solution because it was originally hypothesized that the insert should be placed directly beneath the active area and provide a path directly to the bottom of the cold plate.

4.1.3 Performance of Optimization Methods

The optimization problem was solved using both the NPSOL [17] and DOT [18] packages in the DAKOTA [16] optimization framework. The nonlinearly constrained SQP methods were chosen from each package. Three different types of gradients were used to validate the

Table 4.1 Constrained Optimization Code Performance for Cooling Package Design Problem

	F(x)	C(x)	Steps	F Call	Total Time
DOT/Ext Adj	19.395	-6.501×10^{-4}	14	49	1540.
DOT/Ext Dir	19.235	-1.338×10^{-4}	36	225	3780.
DOT/Int Dir	19.523	1.549×10^{-4}	16	92	3410.
NPSOL/Ext Adj *	2.443	4.675×10^{-2}	6	13	870.
NPSOL/Ext Dir *	10.420	2.847×10^{-2}	2	8	1140.
NPSOL/Int Dir *	10.392	3.139×10^{-2}	2	40	1480.
* failed to converge					

adjoint method and to provide timings for comparison between the "black-box" method and the adjoint method. All numerical derivatives are constructed using forward differences and a step factor of 0.01. The first gradient method is the adjoint method. The second method is the "black-box" method, but run externally to DAKOTA to provide a direct comparison in workload estimates with the adjoint method. The third method is the "black-box" method internal to DAKOTA. The performance times for gradients generated internal and external to DAKOTA should be different because DAKOTA has logic that prevents it from making duplicate function evaluations.

The adjoint method run with the DOT-SQP optimizer reached the optimal solution the fastest, in half the time required than when using gradients generated internally to DAKOTA. Both approaches terminated due to a step-size tolerance of 10^{-4}. The solution with "black-box" gradients, generated externally to DAKOTA, took three times longer than the adjoint approach, but the results cannot be compared directly. The external gradient information is more accurate than the internal DAKOTA gradients or the adjoint gradients because optimizer does not stop due to step-size tolerances. It continues past the point the other two stop to find a better solution. The NPSOL method failed on this problem, unable to find a solution that satisfies the first-order Kuhn-Tucker conditions. Table 4.1 provides a tabulation of the performance for the methods in terms of function value, constraint value, the number of optimization steps, the total number of function evaluations, and the total time for the optimization. The history of the design variables in the iteration process is shown in Figures 4.4 through 4.6.

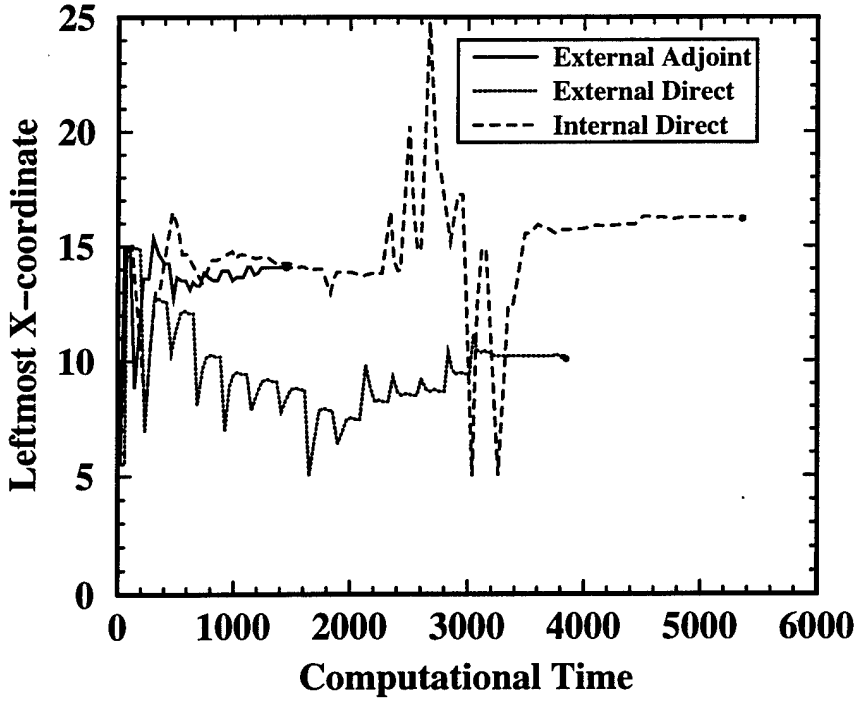


Figure 4.4 History of Leftmost X-coordinate in Iteration Process with DOT-SQP

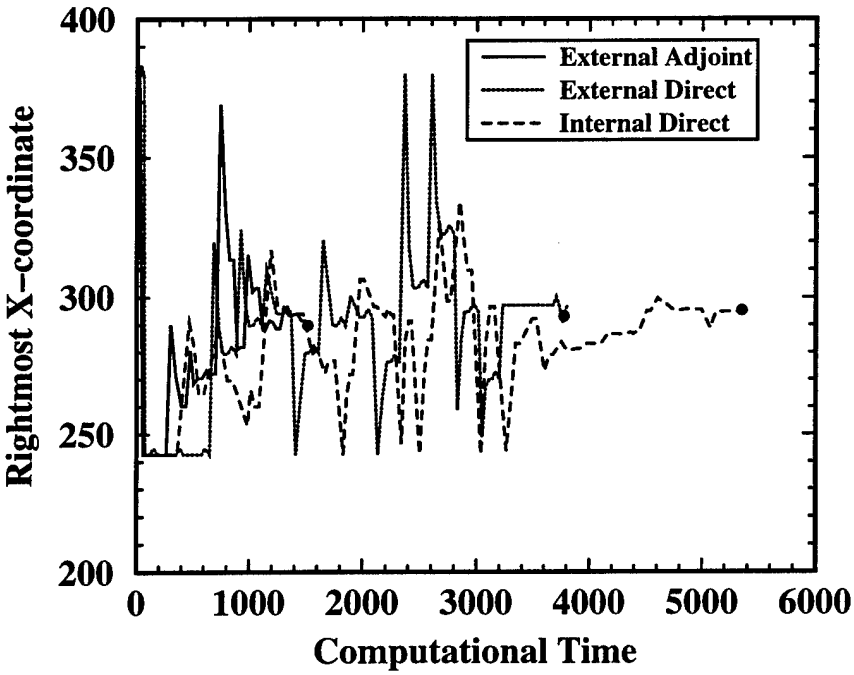


Figure 4.5 History of Rightmost X-coordinate in Iteration Process with DOT-SQP

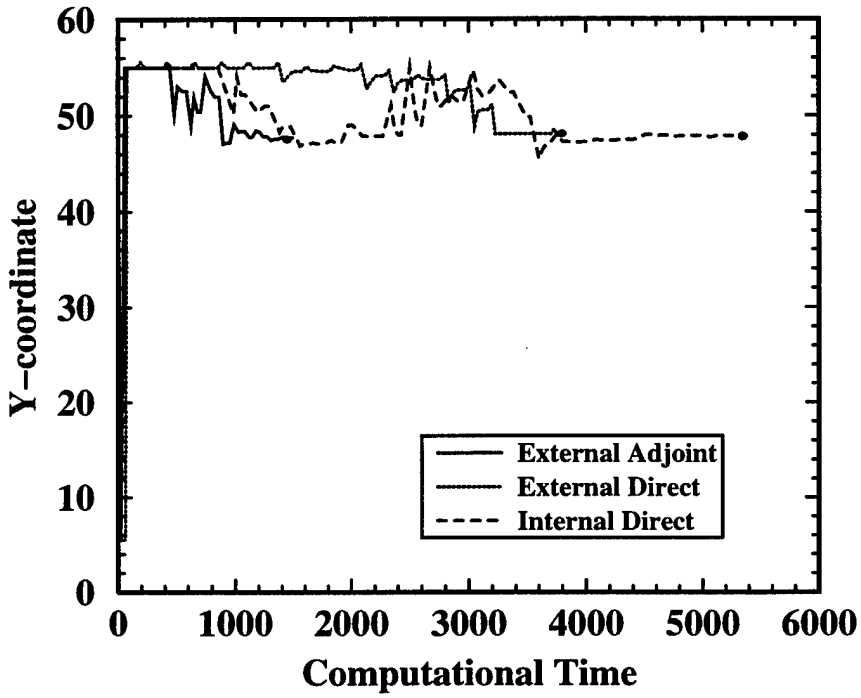


Figure 4.6 History of Y-coordinate in Iteration Process with DOT-SQP

4.2 Flow Channel Design

The channel shape of a horizontal flow chemical vapor deposition reactor can determine the flux uniformity to the wafer surface. The surface flux in a growing laminar boundary layer is nonuniform because the near-wall gradients decrease. The boundary layer thickness can be made uniform by accelerating the flow through a constricted flow passage. Shape optimization for flowing systems is demonstrated by targeting uniform heat flux to the bottom surface of the flow channel.

4.2.1 Optimization Problem

The optimization problem is to minimize the least-squares variation of the bottom-surface heat flux from a target value, given nonlinear constraints on the shape of the flow channel. There are five design variables describing the height of the channel at prescribed axial stations. The bottom surface of the channel is flat. The shape of the top surface is recovered by a tension spline through the five design variables. The convective heat flux to the isothermal channel floor is calculated using CURRENT [12] fluid mechanics code and the ANTIPASTO [19] meshing code.

The problem is exaggerated to provide good visual results by driving the reactor at a higher mass flow rate, thereby requiring a larger acceleration and slope-change to achieve flux uniformity.

The thermodynamic pressure is 1.3×10^5 dynes. The channel walls are fixed at a temperature of 300 K. Nitrogen flows into the reactor at 800 K at 400 cm/s. The channel entrance inflow profile is uniform. The channel entrance region is one inch long and one inch high. The constricted region follows for the next ten inches. The minimum channel height is constrained at a tenth of an inch. The design variables are nonlinearly constrained to be of monotonically decreasing height down the length of the channel.

4.2.2 Design Analysis

The optimal channel shape is shown in Figure 4.7 along with temperature contours. The comparison of the Stanton numbers for the uniform channel and optimized channel are shown in Figure 4.8.

4.2.3 Performance

The calculation of derivative information is simplified because of the relative place of the design variables to the design objective. The objective function is not directly dependent

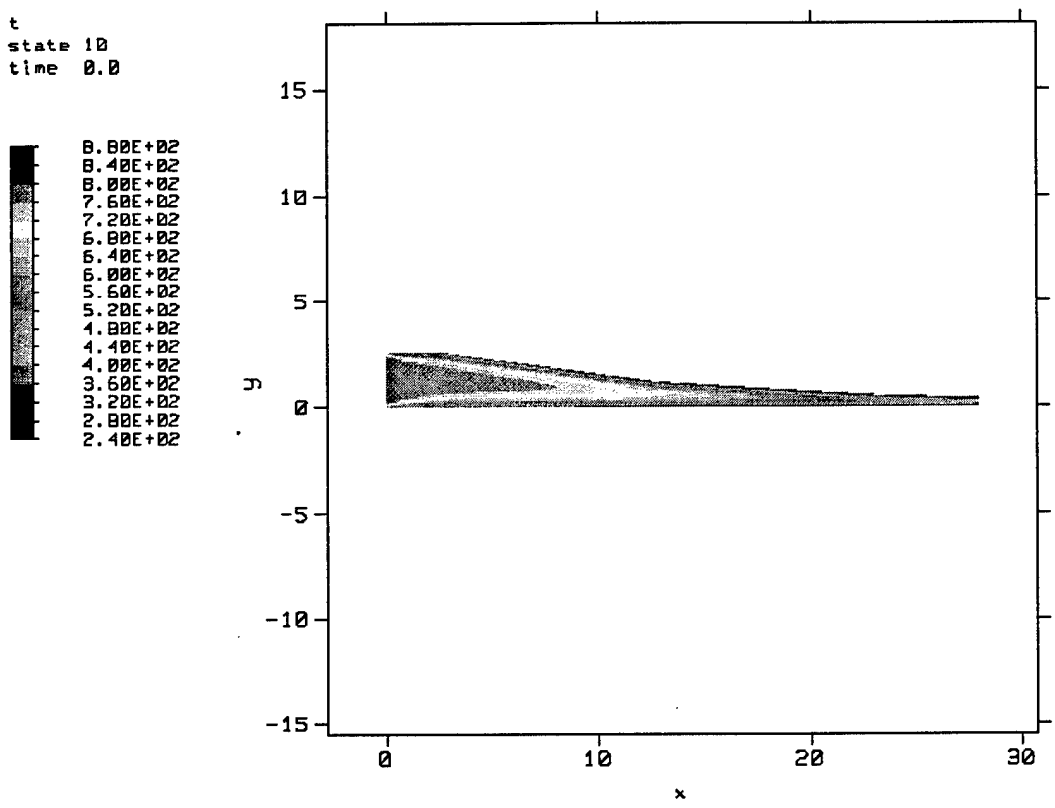


Figure 4.7 Temperature Contours in Optimized Flow Channel

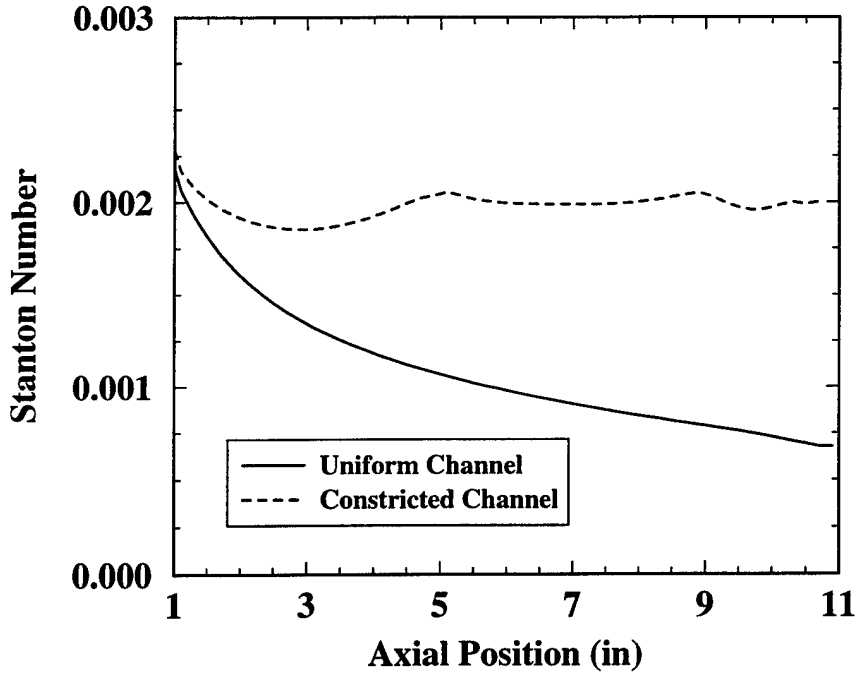


Figure 4.8 Comparison of Stanton Numbers in Uniform and Optimized Channels

on the design variables and the nonlinear constraints do not depend directly on the solution variables. Derivatives of the objective function are simplified–only one set of costate variables are required since the other right-hand-sides of Equation 2.13 are null.

Three versions of the CURRENT code are required. The original code is used to calculate consistent solution variables for a set of design variables and to evaluate the objective function. A second version of the code is required to evaluate and invert the transpose of the heat equation matrix. A third version of the code is required to evaluate the residual of the heat equation and then terminate.

As of the time of this draft, the adjoint-method gradients using the segregated approach are not working. The gradient information is off by a factor of roughly two to three–enough to cause the DOT package to fail after one gradient evaluation.

Chapter 5: Summary

The adjoint method makes use of sensitivity information inherent in the solution algorithm of an analysis code to compute gradient information. Because the sensitivity information already exists, the adjoint method should be cheaper to execute than direct numerical differentiation. The wisdom holds true for large problems, but the cost/benefit numbers for smaller problems get obfuscated by the input parsing and pre- and post-processing overhead of the analysis codes. Also, the adjoint method is beneficial only if the number of design variables is much larger than the number of constraints.

Typically, the information required to execute the adjoint method can be extracted from an analysis code in a relatively non-intrusive manner. But, each code is different and a building general adjoint method software tool does not make sense. The adjoint method can be used in conjunction with the DAKOTA code by supplying the proper information as DAKOTA "analytic gradients".

The adjoint method works best when the full Jacobian matrix is available. Approximate forms of the Jacobian matrix are not accurate to provide useful gradient information. The observation is supported by the example solutions given in this report and research trends in the literature. Gradients computed by the adjoint method are expected to be less accurate than gradients by direct numerical differentiation because of truncation errors in the multiple steps of the adjoint method.

As of the last draft of this report, funding ended before finishing the work on segregated schemes. The approximate adjoint information extracted from the segregated scheme was judged to be too inaccurate, but I think more work is required on the subject. There should be enough sensitivity information available from such a code. The problem is integrating the proper pieces.

References

[1] Moen, C. D. , P. A. Spence and J. C. Meza. "Optimal Heat Transfer Design of Chemical Vapor Deposition Reactors". Technical Report SAND95–8223, Sandia National Laboratories, Livermore, CA, April 1995.

[2] Tong, C. H. , J. C. Meza and C. D. Moen. "Simulation of Equipment Design Optimization in Microelectronics Manufacturing". 30th Annual Simulation Symposium, Society for Computer Simulation/ACM SIGSIM, Atlanta, GA, April 1997.

[3] Moen, C. D. , P. A. Spence , J. C. Meza and T. D. Plantenga. "Automatic Differentiation for Gradient-Based Optimization of Radiatively Heated Microelectronics Manufacturing Equipment". In 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, pages 1167–1175, 1996.

[4] Jameson, A. "Optimum Aerodynamic Design Using CFD and Control Theory". In 12th AIAA Computational Fluid Dynamics Conference, San Diego, CA, pages 926–949, 1995.

[5] Burgreen, G. W. , O. Baysal and M. E. Eleshaky. "Improving the Efficiency of Aerodynamic Shape Optimization". *AIAA Journal*, **32**:69–76, 1994.

[6] Reuther, J. , J. J. Alonso , M. J. Rimlinger and A. Jameson. "Aerodynamic Shape Optimization of Supersonic Aircraft Configurations via an Adjoint Formulation on Parallel Computers". AIAA Paper 96–4045, 6th Multidisciplinary Analysis and Optimization Symposium, Bellevue, WA, September 1996.

[7] Anderson, W. K. and V. Venkatakrishnan. "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation". AIAA Paper 97–0643, Aerospace Sciences Meeting, Reno, NV, January 1997.

[8] Gartling, D. K. and R. E. Hogan. "Coyote II – A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part I: Theoretical Background". Technical Report SAND94–1173, Sandia National Laboratories, Albuquerque, NM, October 1994.

[9] Gartling, D. K. and R. E. Hogan. "Coyote II – A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part II: Users Manual". Technical Report SAND94–1179, Sandia National Laboratories, Albuquerque, NM, October 1994.

[10] Shadid, J. N., H. K. Moffat, S. A. Hutchinson, G. L. Hennigan, K. D. Devine, and A. G. Salinger. "MPSalsa: A Finite Element Computer Program for Reacting Flow Problems, Part 1 – Theoretical Development". Technical Report SAND95–2752, Sandia National Laboratories, Albuquerque, NM, 1996.

[11] Salinger, A., K. Devine, G. Hennigan, H. Moffat, S. Hutchinson, and J. Shadid. "MPSalsa: A Finite Element Computer Program for Reacting Flow Problems, Part 2 – User's Guide". Technical Report SAND96–2331, Sandia National Laboratories, Albuquerque, NM, September 1996.

[12] Winters, W. S. , G. H. Evans and C. D. Moen. "CURRENT–A Computer Code for Modeling Two-Dimensional, Chemically Reacting, Low Mach Number Flows". Technical Report SAND97–8202, Sandia National Laboratories, Livermore, CA, October 1996.

[13] Mason, W. E. "TACO3D – A Three-Dimensional Finite Element Heat Transfer Code". Technical Report SAND83–8212, Sandia National Laboratories, Livermore, CA, April 1983.

[14] Frank, P. D. and G. R. Shubin. "A Comparison of Optimization-Based Approaches for a Model Computational Aerodynamics Design Problem". *Journal of Computational Physics*, **98**:74–89, 1992.

[15] Blacker, T. D. "FASTQ Users Manual, Version 1.2". Technical Report SAND88–1326, Sandia National Laboratories, Albuquerque, NM, 1988.

[16] Eldred, M. S, , W. E. Hart , W. J. Bohnhoff , V. J. Romero , S. A. Hutchinson and A. G. Salinger. "Utilizing Object-Oriented Design to Build Advanced Optimization Strategies with Generic Implementation". In 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, pages 1568–1582, 1996.

[17] Gill, P. E. , W. Murray , M. A. Saunders and M. H. Wright. "Users Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming". Technical Report SOL 86-2, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA, January 1986.

[18] "DOT Users Manual, Version 4.20". Vanderplaats Research and Development, Inc., Colorado Springs, CO, 1995.

[19] Nielan, P. E., K. J. Perano, and W. E. Mason. "ANTIPASTO: An Interactive Mesh Generator and Preprocessor for Two-Dimensional Analysis Programs". Technical Report SAND90–8203, Sandia National Laboratories, Livermore, CA, 1990.

APPENDIX E

Automatic Differentiation for Gradient-Based Optimization of Radiatively Heated Microelectronics Manufacturing Equipment

conf cycled separately.

DISTRIBUTION:

1 Eric R. Ponslet
HYTEC, Inc.
110 Eastgate Dr.
Los Alamos, NM 87544

1 David C. Zimmerman
Dept. of Mechanical Engineering
University of Houston
Houston, TX 77204-4792

1 MS 0151 R. D. Skocypec, 9002
1 MS 0188 LDRD Office, 4523
1 MS 0321 W. J. Camp, 9200
20 MS 0439 M. S. Eldred, 9234
1 MS 0439 D. R. Martinez, 9234
1 MS 0439 R. E. Rhea, 9234
1 MS 0439 W. R. Witkowski, 9234
1 MS 0443 H. S. Morgan, 9117
1 MS 0503 R. B. Diegle, 2338
1 MS 0557 C. C. O'Gorman, 9741
1 MS 0557 T. L. Paez, 9741
1 MS 0557 T. W. Simmermacher, 9741
1 MS 0746 D. G. Robinson, 6411
1 MS 0746 L. P. Swiler, 6411
1 MS 0819 J. S. Peery, 9231
1 MS 0819 T. G. Trucano, 9231
1 MS 0820 P. Yarrington, 9232
1 MS 0826 K. S. Chen, 9111
1 MS 0826 W. L. Hermina, 9111
1 MS 0826 P. R. Schunk, 9111
1 MS 0828 T. C. Bickel, 9101
1 MS 0828 J. H. Biffle, 9103
1 MS 0828 R. K. Thomas, 9104
1 MS 0834 M. L. Hobbs, 9112
1 MS 0835 B. L. Bainbridge, 9113
1 MS 0835 B. F. Blackwell, 9113
1 MS 0835 V. J. Romero, 9113
1 MS 0841 P. J. Hommert, 9100
1 MS 1010 G. R. Eisler, 9622
1 MS 1110 W. E. Hart, 9222
1 MS 1110 D. E. Womble, 9222
1 MS 1111 S. A. Hutchinson, 9221
1 MS 1111 A. G. Salinger, 9221
1 MS 1174 D. E. Outka, 2526
1 MS 1179 W. J. Bohnhoff, 9341
1 MS 9042 C. D. Moen, 8345
1 MS 9042 P. A. Spence, 8345
1 MS 9214 T. D. Plantenga, 8950
1 MS 9214 J. C. Meza, 8950
1 MS 9402 C. M. Hartwig, 8701

1 MS 9018 Central Technical Files, 8940-2
2 MS 0899 Technical Library, 4916
2 MS 0619 Review and Approval Desk, 12690
For DOE/OSTI

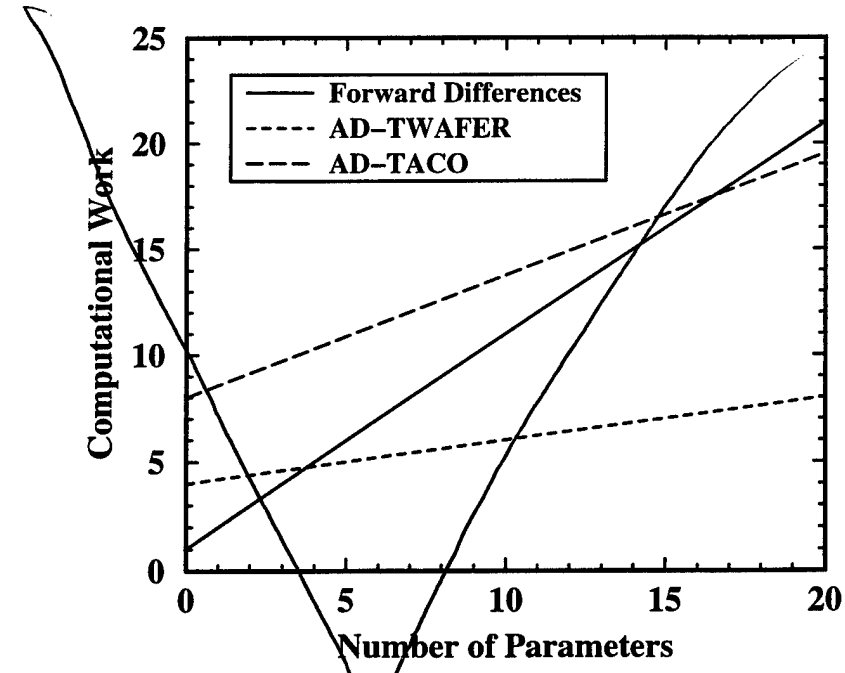


Fig. 7 Analytic Gradients Accelerate Convergence of Optimization Algorithms

Table 3 Function-Call Timings for the TACO Code

	5 Zone	9 Zone	15 Zone	25 Zone
TACO/Matrix	0.1	0.1	0.1	0.1
TACO/Force Vector	0.17	0.17	0.17	0.17
TACO/Time Step	5.8	5.8	5.8	5.8
AD_TACO/Matrix	0.96	1.08	1.36	1.89
AD_TACO/Force Vector	1.35	1.75	2.34	3.12
AD_TACO/Time Step	62.1	79.4	96.3	130.3
Timings in CPU seconds on SGI Power Challenge				

Report Number (14) SAND--98-0340

Publ. Date (11) 199802

Sponsor Code (18) DOE/DP, XF

UC Category (19) UC-705, DOE/ER

DOE