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.

[*]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. This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

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-

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.

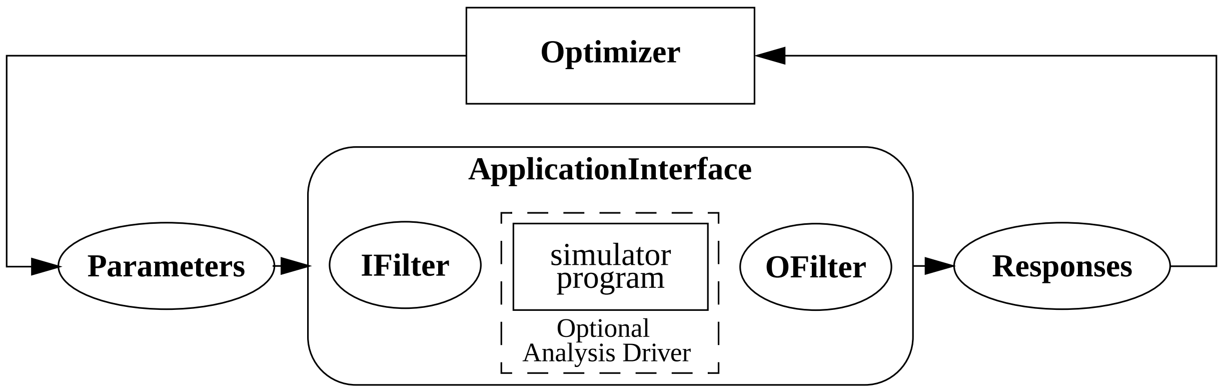

Figure 1. Application interface conceptualization.

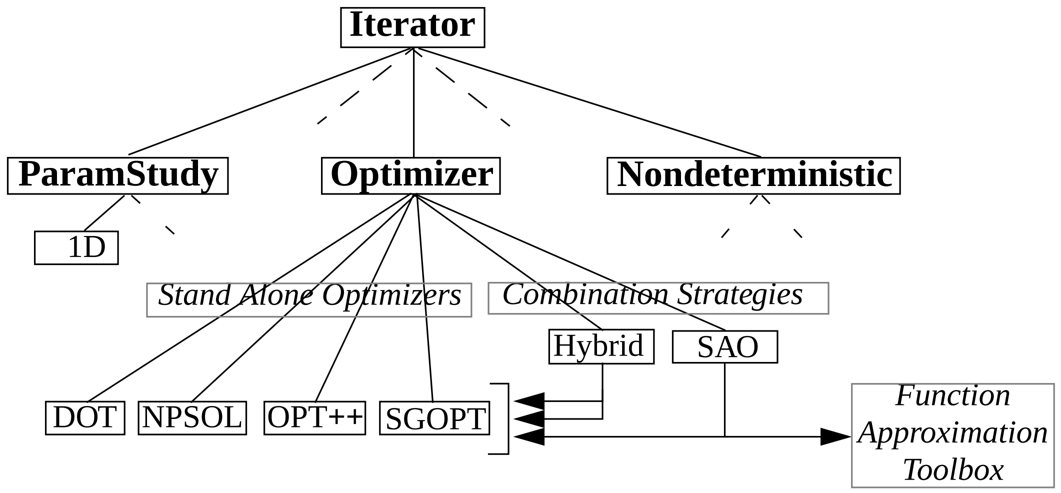

Figure 2. Iterator hierarchy showing broad iteration capabilities:
A) **ParamStudy**: for mapping response variations with respect to model parameters.
B) **Optimizer**: for numerical optimization studies.
C) **Nondeterministic**: for assessing the effect of modeling uncertainties on responses.

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^{O} 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 ($0.5 \leq r \leq 5.8$, $-2.9 \leq x \leq 2.9$). Early optimization studies solved this 2 parameter problem; later studies added the y degree of freedom in fire location as a third parameter ($0.0 \leq y \leq 2.9$). 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

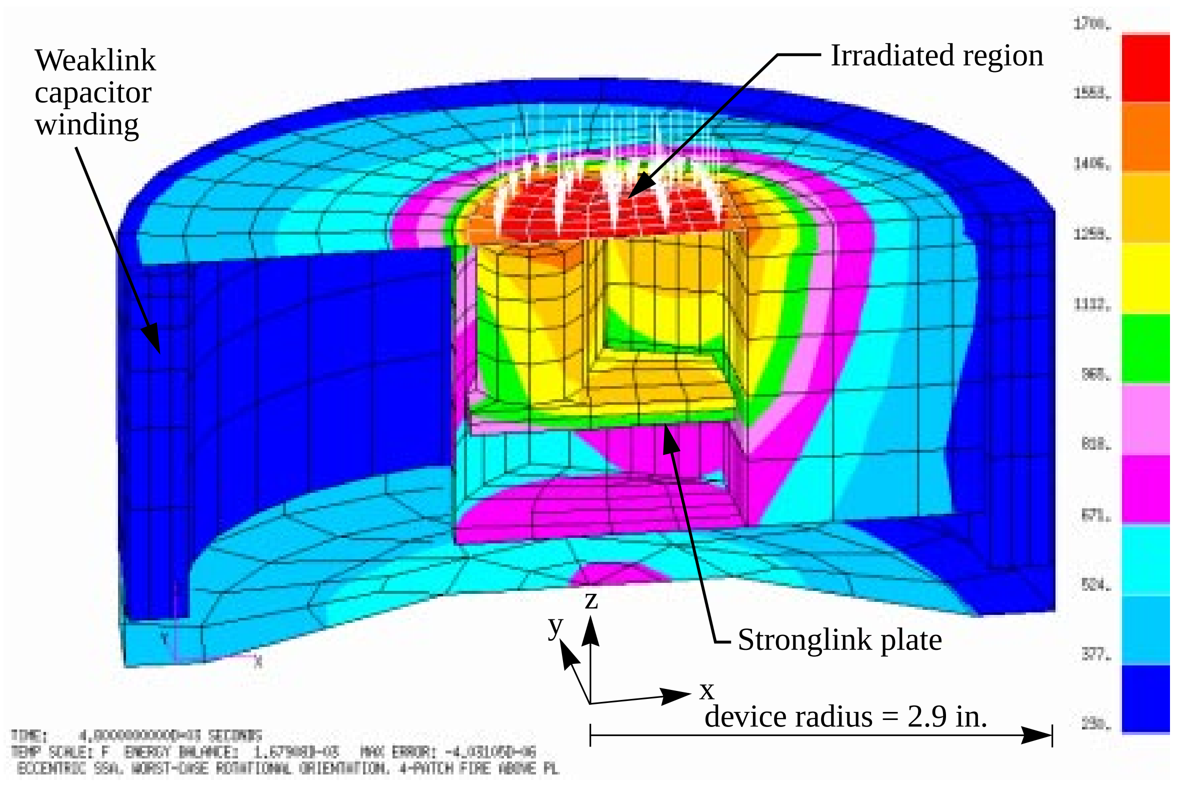
Weaklink capacitor winding
Irradiated region
Stronglink plate
z
y
x
device radius = 2.9 in.
1700.
1553.
1406.
1259.
1112.
965.
818.
671.
524.
377.
230.
TIME: 4.00000000000D+03 SECONDS
TEMP SCALE: F ENERGY BALANCE: 1.67908D-03 MAX ERROR: -4.03105D-06
ECCENTRIC SSA, WORST-CASE ROTATIONAL ORIENTATION, 4-PATCH FIRE ABOVE PL

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

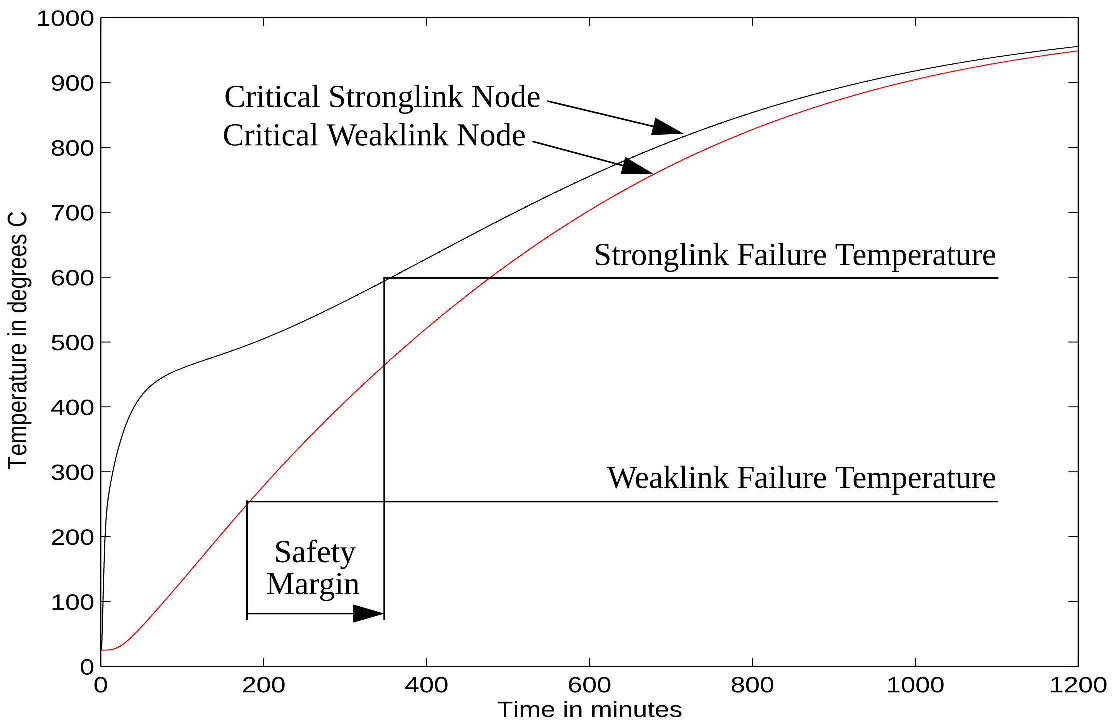

Figure 4. Temperature histories of critical stronglink and weaklink nodes.

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).

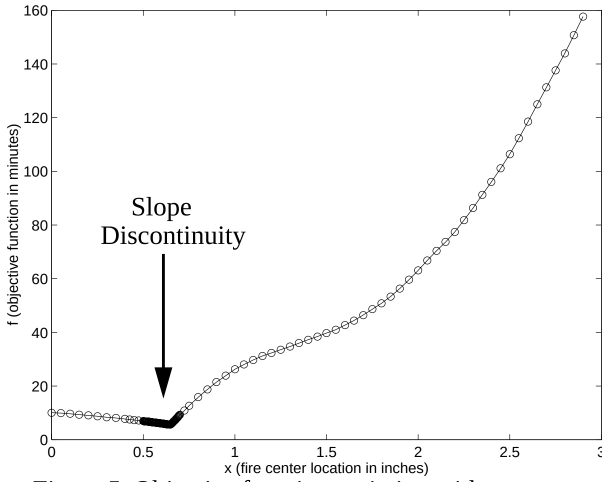

Figure 5. Objective function variation with respect to fire center location (x) for r=1.89.

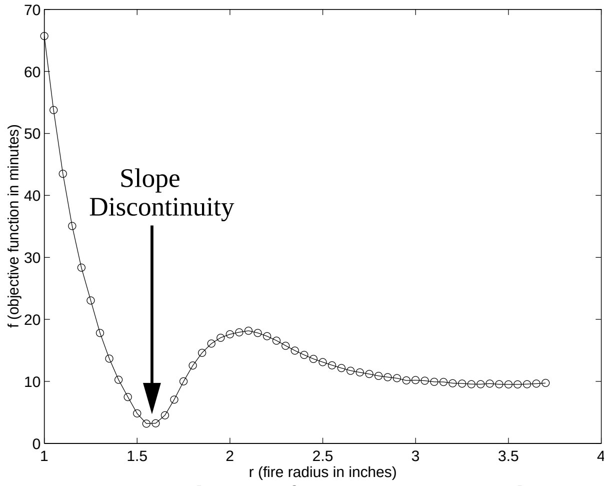

Figure 6. Objective function variation with respect to fire radius (r) for x=0.8.

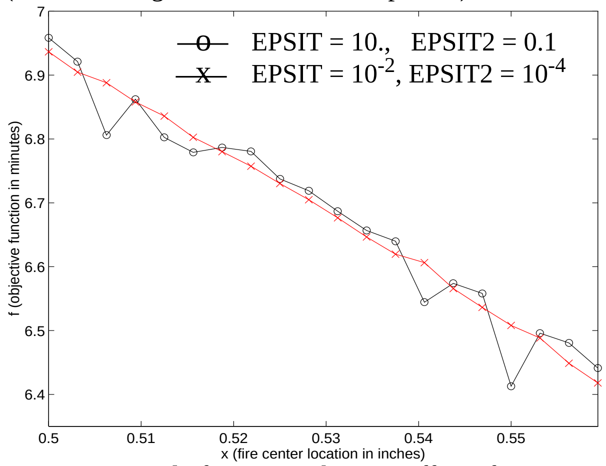

Figure 7. Detail of Figure 5 showing effect of QTRAN convergence tolerances on design space nonsmoothness.

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).

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,

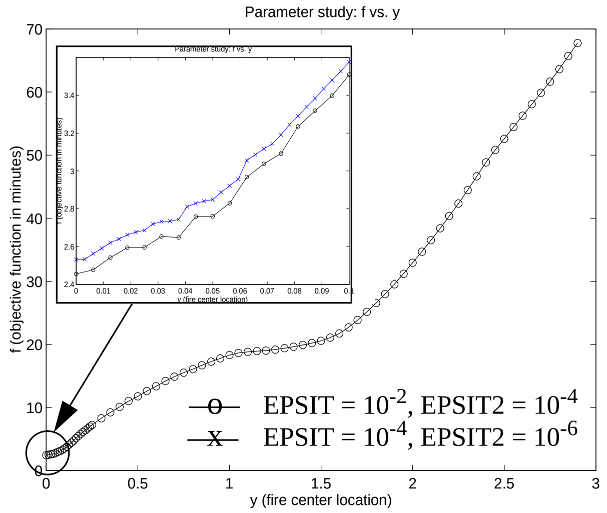

Figure 8. Objective function variation with respect to fire location (y) for r=1.6204 and x=0.78205. Detail shows local nonsmoothness for 0≤y≤0.1

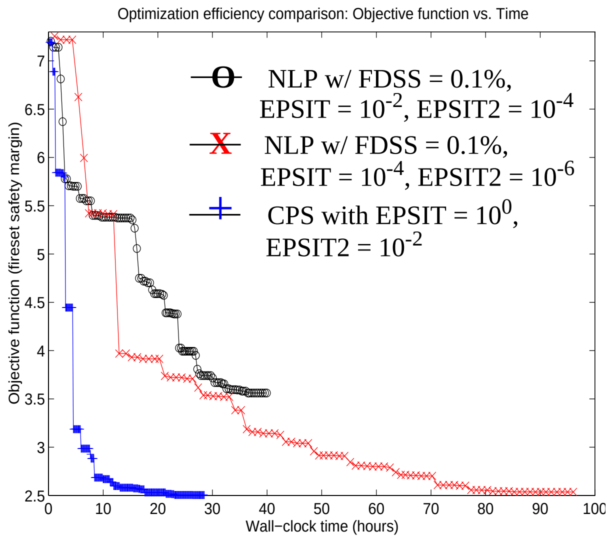

Figure 9. Optimization history comparison: Best objective function value vs. wall-clock time in hours

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

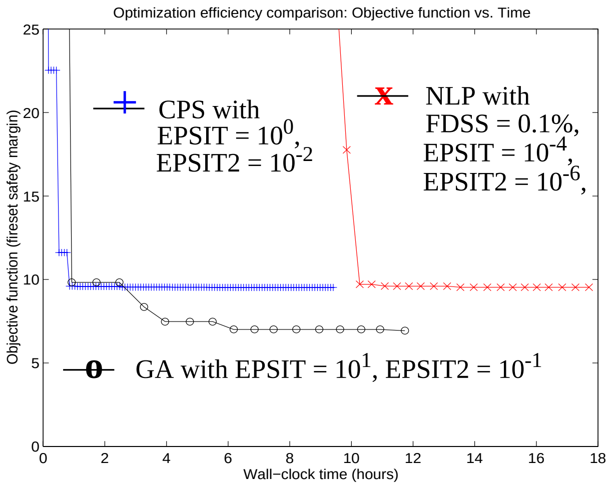

Figure 10. Optimization history for GA runs compared with CPS and NLP starting from (r,x,y)=(1.9, 2.1, 0.0).

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

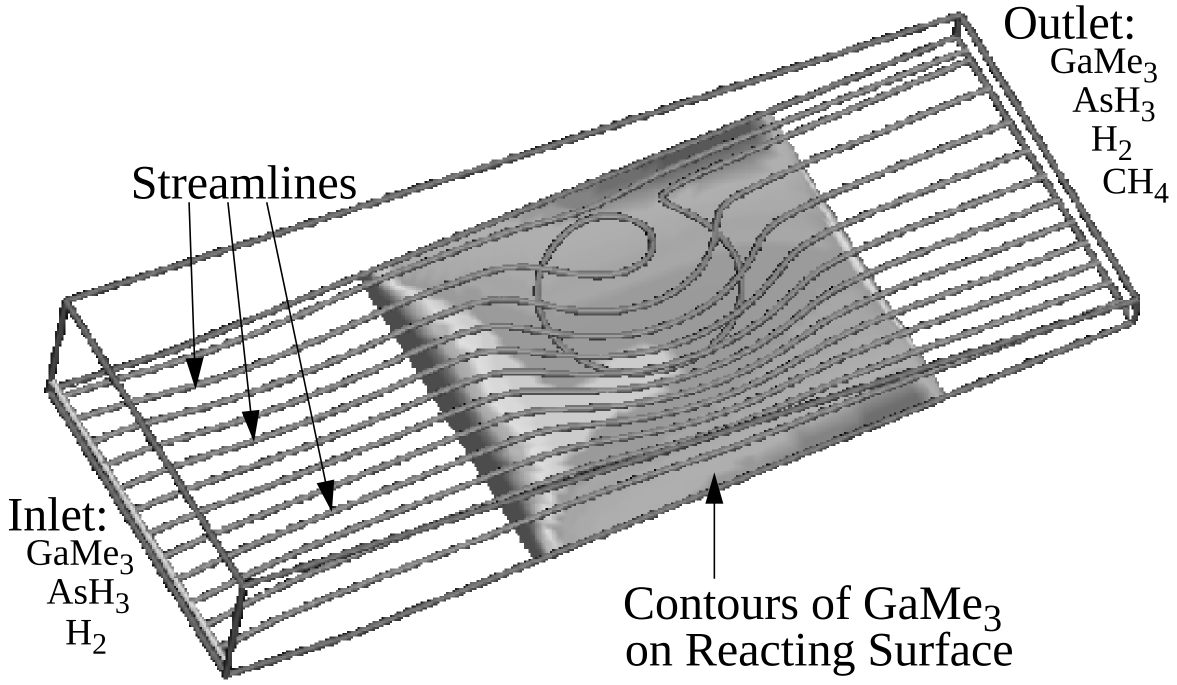

Figure 11. Deposition of Gallium Arsenide in a horizontal CVD reactor with tilted susceptor.

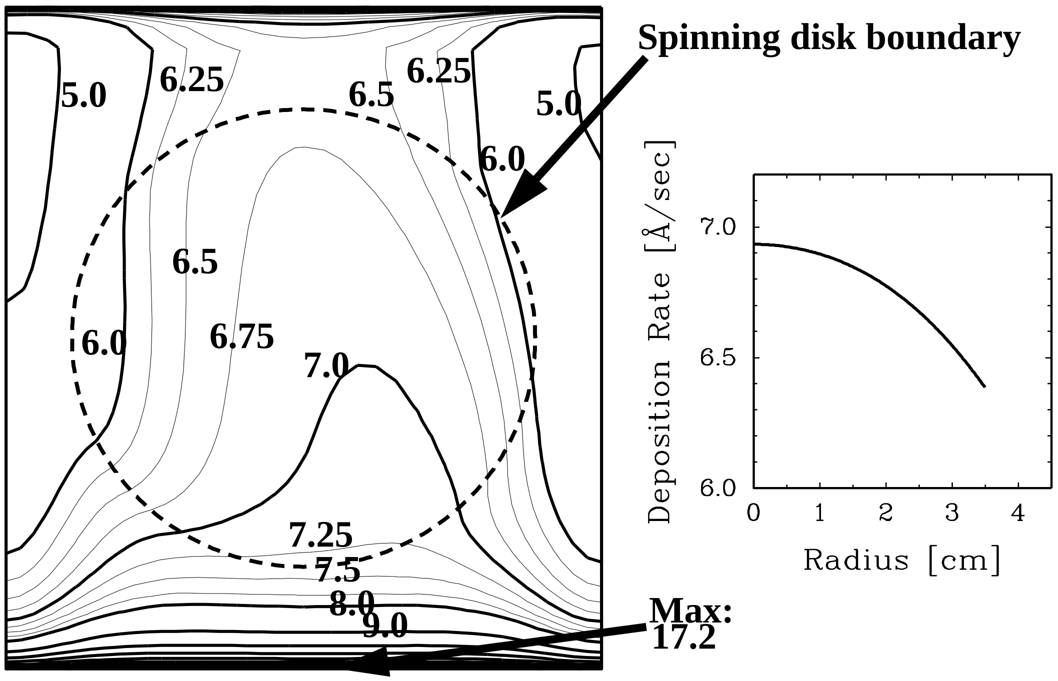

Figure 12. Contours of the deposition rate of GaAs over the entire reactive surface (left) and time-averaged deposition rate over the spinning disk (right). Attempting to maximize the growth rate while maintaining required uniformity over the disk is a major component of the optimization of this reactor.

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

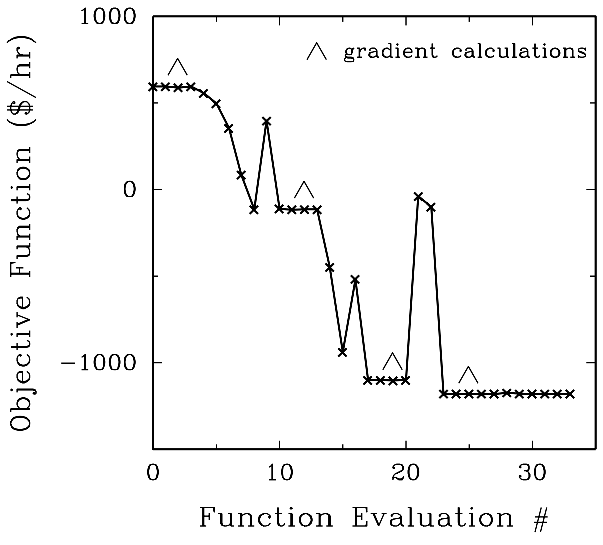

Figure 13. Objective function history for a 3 parameter optimization of a CVD reactor on a coarse mesh. Each conjugate gradient iteration began with a gradient calculation as marked.

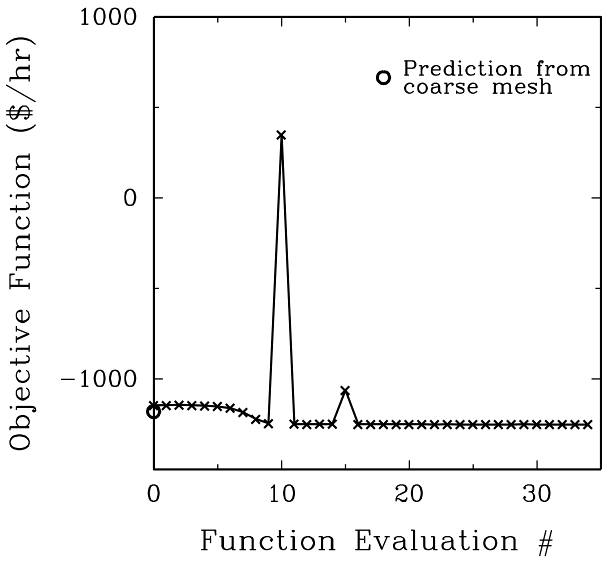

Figure 14. Objective function history for CVD reactor optimization on the fine mesh with initial guess from the coarse mesh converged solution. Each function evaluation required the solution of 366480 unknowns, which were solved on 512 processors of the Intel Paragon in 5-8 minutes each.

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

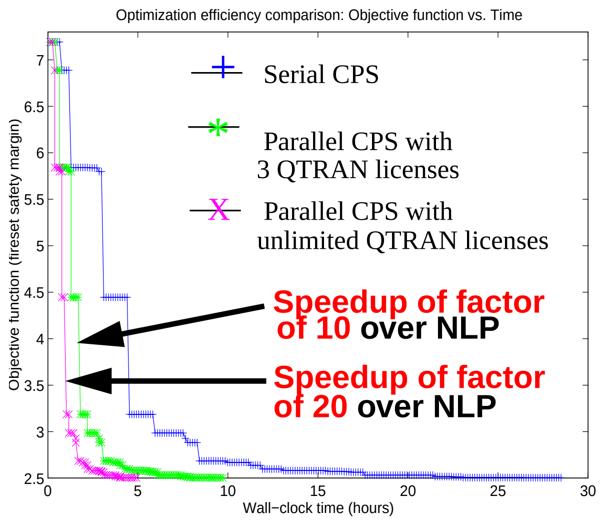

Figure 15. Optimization history comparison: Best objective function value vs. wall-clock time in hours

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

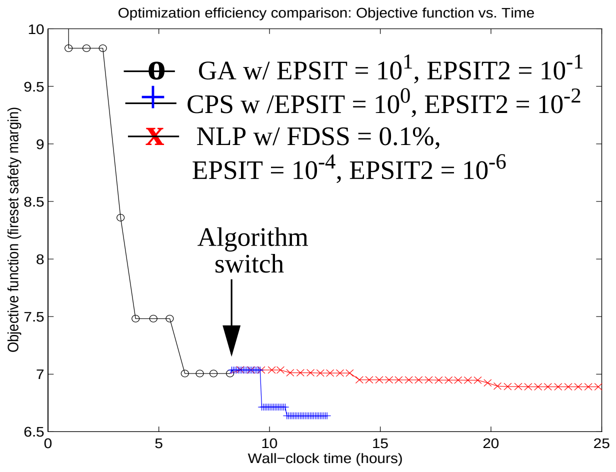

Figure 16. Optimization history comparison for hybrid GA/NLP and GA/CPS strategies: Best objective function value vs. wall-clock time in hours

(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

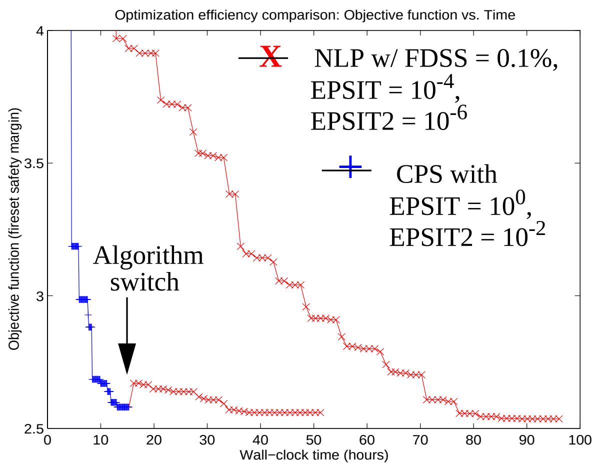

Figure 17. Optimization history comparison of NLP and CPS/NLP hybrid: Best objective function value vs. wall-clock time in hours

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.