AIAA 2010-9325

13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference
13–15 September, 2010, Fort Worth, Texas, United States

A New Approach to Multidisciplinary Design Optimization via Internal Decomposition

Andrew B. Lambe,*
University of Toronto Institute for Aerospace Studies, Toronto, ON, Canada
Joaquim R. R. A. Martins†
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI

Traditional approaches to MDO problem decomposition have shown poor performance when solving problems with strong interactions between disciplines. We present a new decomposition strategy for MDO that aims to overcome this difficulty. The method, called Block Approximation with Krylov Refinement, or BAKR, decomposes the solution of the linear system present at each iteration of an interior point algorithm. By decomposing the linear system inside the optimization algorithm, rather than the original design problem, we maintain the strong global and local convergence properties of the interior point algorithm while reducing the overall computational cost of the solution. Preliminary test results show reductions in both computational cost and the number of function evaluations, demonstrating strong potential for future application in large MDO problems.

I. Introduction

A multidisciplinary design optimization (MDO) architecture is a particular strategy for organizing analysis software and optimization software in order to solve a design optimization problem. A large number of architectures exist in the literature for solving MDO problems with different characteristics.[1–7] Selecting an appropriate architecture for a particular MDO problem can significantly reduce the computational time needed for the solution of that problem.

While many architectures have been developed, the most intensively studied have been those we call *distributed* architectures. These are architectures that employ decomposition to partition the design optimization problem and distribute the work to many processors in a computer network. This approach is advantageous because it resembles existing industrial design practices, where the design of a complex system is partitioned and each aspect is distributed to a different group in the organization. The similarity of these approaches simplifies the implementation of distributed architectures in industry. At the same time, exploiting problem structure through decomposition allows very large design problems to be solved with relatively modest computing facilities in a reasonable amount of time. Unfortunately, the performance of distributed architectures on practical problems is still largely unknown. Studies that are available indicate that optimization with distributed architectures is frequently less efficient than with *monolithic* architectures, those that do not employ decomposition.[8–10] In particular, distributed architectures typically require many more disciplinary analysis calls than monolithic architectures on tested problems. Thus, the search for distributed architectures with good performance in practice continues.

Existing distributed architectures implement decomposition in one of two ways: multilevel decomposition and penalty decomposition. In multilevel decomposition, the MDO problem is decomposed into at least two levels of subproblems, possibly using an established hierarchy. (If there is no established hierarchy, the MDO problem is divided into system and discipline subproblems.) At each level in this decomposition scheme, local

*Ph.D. Candidate, AIAA Student Member
†Associate Professor, AIAA Senior Member

design variables and constraints are resolved at the discipline level, while shared (global) design variables, shared design constraints, and interdisciplinary consistency constraints are resovled at the system level. In order to obtain an optimal design, post-optimality sensitivity analysis is required at the upper levels so that the optimality of the lower level problems is preserved during each system iteration. Examples of multilevel decomposition architectures include collaborative optimization,[11] and bi-level integrated system synthesis.[12]
In the penalty decomposition approach, no set hierarchy of problems is created. Instead, copies of the global variables, objectives, and constraints exist in each disciplinary subproblem. Interdisciplinary consistency is enforced at optimality through the use of penalty functions, and the penalty weights are updated after all subproblems have been solved for the current weight values. Examples of penalty decomposition architectures include enhanced collaborative optimization[13] and analytical target cascading.[14] Note that analytical target cascading may be employed with either quadratic,[15] Lagrangian,[16] or augmented Lagrangian[17,18] penalty functions.
In both distributed architecture classes, techniques used for traditional optimization of large systems are employed. However, multilevel optimization and penalty-based optimization are not the most advanced techniques available for nonlinear optimization. The algorithms that are accepted as being the most effective for general-purpose nonlinear optimization are sequential quadratic programming (SQP) and interior point (IP) algorithms. To our knowledge, techniques from these optimization algorithms have yet to be applied to a distributed architecture to decompose the optimization problem. However, applying SQP and IP algorithms in monolithic architectures in such a way that coarse-grained parallel processing is more effectively utilized may point towards more effective distributed architectures.
In this work, we present the first steps in applying this last approach to MDO problems. Instead of decomposing the original optimization problem for solution, we decompose the solution to the linear systems generated by IP auxiliary problems. The decomposition is achieved by using block-separable approximations of the Jacobian and Hessian matrices to compute a fast approximate solution to the linear system. If necessary, this approximation is refined to sufficient accuracy using a Krylov subspace method. We name this approach Block Approximation with Krylov Refinement, or BAKR.
Note that because we maintain only a single optimization problem at each iteration, the resulting MDO architecture is *not* a distributed architecture in the strict sense. However, compared to a standard monolithic architecture implemenation, this approach is much more amenable to coarse-grained parallel processing and more able to exploit optimization problem structure. A similar approach may also be used to decompose the solution of linear systems generated by SQP auxiliary problems. Strategies that employ decomposition within an optimization algorithm, rather than decomposition of the original problem, are called *internal decomposition* methods.[19] Related work on internal decomposition with interior point methods is given by Laird and Biegler,[19] Zavala et al.,[20] and Gondzio and Grothey.[21,22]
The specific algorithm discussed in this work is superficially similar to an algorithm first described by Conejo et al.,[23] later termed "Optimality Condition Decomposition."[24] However, our algorithm is different in three important respects. First, we make use of approximate second-order information at each iteration, while Conejo's algorithm uses exact information. Second, our algorithm employs a merit-function-based line search strategy to ensure global convergence on nonconvex problems while Conejo's algorithm uses no global convergence strategy. Finally, our algorithm is able to handle variable bounds and general inequality constraints while Conejo's algorithm handles only equality constraints.
The remainder of this paper is organized as follows: Section II provides further details on the motivation behind the internal decomposition approach; Section III details the optimization algorithm and how if differs from traditional interior point methods; Section IV discusses preliminary results on test problems; finally, Section V summarizes the work and discusses future research directions.

II. Motivation

In nonlinear optimization, it is often impossible to directly find a feasible local minimum of the objective function. Instead, given an initial point, we solve a sequence of simpler optimization problems to generate an appropriate sequence of new points that converges to a local minimum. Cohen[25,26] termed these simpler optimization problems "auxiliary problems", and gave conditions under which the auxiliary problems converge to a minimum point. In the case of an SQP algorithm, this auxiliary problem takes the form of a quadratic objective function with linear equality and inequality constraints. The auxiliary problem in a general IP algorithm is similar to that in an SQP algorithm, but slack variables are added to convert the inequality

constraints to equalities and all variable bounds are satisfied by adding logarithmic barrier functions to the objective.

Cohen also showed that it is possible to decompose the auxiliary problem into a series of auxiliary subproblems as long as the auxiliary problem has a separable structure. Separability, in this case, means that the auxiliary problem must be the sum of independent auxiliary subproblems. The trade-off for this separable structure, however, is a potential loss of performance. Patriksson[27] showed that all decomposition algorithms developed under Cohen's framework can have no better than a linear rate of convergence. The linear rate of convergence is defined by the relation

$$\frac{||x^{(k+1)} - x^*||}{||x^{(k)} - x^*||} \leq \delta,$$

where $x^{(k)}$ is the current iterate, x^* is the optimal solution, k is a large iteration number, and δ strictly between zero and one.[28] This is the same theoretical convergence rate as the steepest-descent algorithm and block-coordinate descent methods. (Indeed, it is quite straightforward to develop these algorithms under Cohen's framework.) On strongly coupled nonlinear problems, such as those found in MDO, these algorithms perform poorly.

In contrast, SQP and IP algorithms are built around Newton or quasi-Newton methods, both of which exhibit a superlinear convergence rate near a local minimum. Mathematically, the superlinear convergence rate is defined by

$$\lim_{k\to\infty} \frac{||x^{(k+1)} - x^*||}{||x^{(k)} - x^*||} = 0.$$

For large values of k, the superlinearly convergent sequence will always approach optimality faster than the linearly convergent one.[28] This property translates into fewer iterations on nonlinear optimization problems and fewer disciplinary simulations on MDO problems, especially near the optimum. The higher rate of convergence is due to the algorithm using complete curvature and constraint information. (In the case of a quasi-Newton method, the curvature information is approximate but gets more accurate as the algorithm approaches optimality.) Some, if not all, of this information is truncated through the decomposition algorithms described by Cohen's framework. The penalty for the higher convergence rate is a higher cost per algorithm iteration and an inability to decompose the auxiliary problem easily for a general problem.

Fortunately, we may exploit two key observations in the solution of large optimization problems. The first observation is that almost all large problems exhibit some type of special structure in the objective and constraint definitions. In the case of MDO, the individual discipline feasible (IDF) and all-at-once (AAO) problem definitions exhibit special structure.[29] (The equivalent multidisciplinary feasible (MDF) and simultaneous analysis and design (SAND) problem definitions may exhibit special structure in specific problem instances but do not in the general case.) The IDF formulation for a problem with N disciplines is given by

$$\begin{aligned}
\text{minimize} \quad & f_0\left(x_0, y^t\right) + \sum_{i=1}^{N} f_i\left(x_0, x_i, y_i\left(x_0, x_i, y^t_{j\neq i}\right)\right) \\
\text{with respect to} \quad & x_0, x_i, y^t_i \quad i = 1, ..., N \\
\text{subject to} \quad & c_0\left(x_0, y^t\right) \geq 0 \\
& c_i\left(x_0, x_i, y_i\left(x_0, x_i, y^t_{j\neq i}\right)\right) \geq 0 \quad i = 1, ..., N \\
& y^t_i - y_i\left(x_0, x_i, y^t_{j\neq i}\right) = 0 \quad i = 1, ..., N
\end{aligned} \tag{1}$$

where the vectors x_i are design variables local to discipline i, x_0 are shared design variables, y_i are the output coupling variables from discipline i, y^t_i are the input (target) coupling variables to disciplines other than i, the functions f_0 and f_i are shared and local design objectives, and c_0 and c_i are shared and local design constraints. Note that we have assumed that the MDO problem is quasiseparable, in the sense defined by Haftka and Watson.[30] In our experience, we have not encountered a design problem that cannot be converted into this general statement. In the AAO formulation, the output coupling variables y_i are under direct optimizer control and the governing equations used to compute them are treated as optimization

constraints. The AAO problem formulation equivalent to (1) is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x_0, y^t\right) + \sum_{i=1}^{N} f_i\left(x_0, x_i, y_i\right)) \\ \text{with respect to} \quad & x_0, x_i, y_i, y_i^t \quad i = 1, ..., N \\ \text{subject to} \quad & c_0\left(x_0, y^t\right) \geq 0 \\ & c_i\left(x_0, x_i, y_i\right) \geq 0 \quad i = 1, ..., N \\ & y_i^t - y_i = 0 \quad i = 1, ..., N \\ & \mathcal{R}_i\left(x_0, x_i, y_i, y_{j \neq i}^t\right) = 0 \quad i = 1, ..., N \end{aligned} \tag{2}$$

where $\mathcal{R}_i$ represents residuals of the governing equations present in the i^{th} discipline analysis. In specific instances, these groups of equations may also exhibit their own structure, which can be further exploited in the solution algorithm. Both IDF and AAO formulations are referred to as "complicating variables" problems in the optimization literature.[23] If the shared (complicating) variables x_0 and y^t were fixed, both the IDF and AAO problems would be completely separable due to the disciplinary objective and constraint functions depending on only local variables.

The second key observation is that we do not need to solve the auxiliary problems exactly to achieve superlinear convergence of the algorithm near an optimum. Indeed, when we are far away from an optimal solution, it may be advantageous to obtain a low-cost approximation to the Newton or quasi-Newton direction rather than compute it exactly. As we approach an optimal solution, the Newton or quasi-Newton direction is computed with increasing accuracy to recover superlinear convergence. This basic result was proven by Dembo et al.[31] for Newton methods and by Steihaug[32] for quasi-Newton methods. We will exploit both problem structure and inexact direction computations in our decomposition framework.

III. Algorithm

We now outline our algorithm for an interior point method on a complicating variables problem. For clarity, we describe the algorithm in terms of a general complicating variables problem statement and note the close connection with the IDF and AAO problems (1) and (2). A complicating variables problem with three distinct local variable groups can be described by

$$\begin{aligned} \text{minimize} \quad & f(x_0) + f(x_0, x_1) + f_2(x_0, x_2) + f_3(x_0, x_3) \\ \text{with respect to} \quad & x_0, x_1, x_2, x_3 \\ \text{subject to} \quad & c_0(x_0) \leq 0 \\ & c_1(x_0, x_1) \leq 0 \\ & c_2(x_0, x_2) \leq 0 \\ & c_3(x_0, x_3) \leq 0 \\ & x_{Li} \leq x_i \leq x_{Ui} \quad \text{for } i = 0, 1, 2, 3. \end{aligned} \tag{3}$$

Note that this problem and the subsequent algorithm can be easily extended to the case of N distinct local variable groups. To find the solution to Problem (3), we first define the Lagrangian function

$$\begin{aligned} \mathcal{L}(x, \lambda, s) = & f_0(x_0) + \sum_{i=1}^{3} f_i(x_0, x_i) + \lambda_0^T \left(c_0(x_0) + s_0\right) + \sum_{i=1}^{3} \lambda_i^T \left(c_i(x_0, x_i) + s_i\right) \\ & - \sum_{k=0}^{3} \left(\nu_{kL}^T (x_k - x_{kL}) + \nu_{kU}^T (x_{kU} - x_k)\right) \end{aligned}$$

where λ_i are the constraint Lagrange multiplier vectors, s_i are the slack variable vectors, and ν_{kL} and ν_{kU} are the variable bound Lagrange multiplier vectors. For a local minimum of (3) to exist at a point $(x^*, \lambda^*, \nu_L^*, \nu_U^*, s^*)^T$, we require $\nabla_x \mathcal{L} = 0$, $\nabla_\lambda \mathcal{L} = 0$, $\nabla_{\nu_L} \mathcal{L} = 0$, $\nabla_{\nu_U} \mathcal{L} = 0$, every elementwise product $\lambda_j^* s_j^* = 0$, $\nu_{Lj}^*(x_j^* - x_{Lj}) = 0$, $\nu_{Uj}^*(x_j^* - x_{Uj}) = 0$, all decision variables are within their upper and lower bounds, and all multipliers and slack variables are nonnegative. These are the Karush–Kuhn–Tucker (KKT)

necessary optimality conditions. The KKT conditions involving the elementwise products and variable bounds are referred to as the complementarity conditions.

In an interior point algorithm, the KKT conditions are treated as a nonlinear system and solved using a Newton method.[33] A perturbation term is attached to the KKT residuals to ensure that, given initial data within bounds, the bounds are always strictly satisfied. In this case, the auxiliary problem is the solution of the linear system

$$\begin{bmatrix} H & J^T & 0 & -I & I \\ J^T & 0 & I & 0 & 0 \\ 0 & S & \Lambda & 0 & 0 \\ N_L & 0 & 0 & (X - X_L) & 0 \\ -N_U & 0 & 0 & 0 & (X_U - X) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \\ \Delta\nu_L \\ \Delta\nu_U \end{bmatrix} = \begin{bmatrix} -\nabla_x \mathcal{L} \\ -(c(x) + s) \\ -\Lambda S e + \sigma\mu e \\ -(X - X_L)N_L e + \sigma\mu e \\ -(X_U - X)N_U e + \sigma\mu e \end{bmatrix}, \tag{4}$$

where I is the identity matrix, H is the Hessian of the Lagrangian, J is the constraint Jacobian, other capital letters denote a diagonal matrix whose diagonal elements are the elements of the corresponding lower-case vector, and e is a vector of ones. The perturbation term consistst of the scalars μ and σ, where μ is the average complementarity residual, and σ is a number between zero and one, referred to as the centering parameter.[33] If all λ, ν and s values are initially chosen to be strictly positive and all x are chosen strictly between their bounds, they will all remain feasible for the whole algorithm as the complementarity conditions approach zero.

Note that we did not include equality constraints in our model problem (3). However, equalities are present in both the IDF problem (1) and the AAO problem (2). This was done simply to highlight the treatment of inequalities and bounds within the interior point algorithm. Equalities can be included explicitly in system (4) as Jacobian rows that do not have an associated slack variable.

While the system (4) is very large, it is also specially structured, with many diagonal submatrices. This structure can be exploited to transform (4) into a block 2×2 system with modest effort.[33,34] The resulting system is

$$\begin{bmatrix} H + \Theta_L + \Theta_U & J^T \\ J & -D \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -\nabla f - J^T\lambda - \sigma\mu\left((X_U - X)^{-1} - (X - X_L)^{-1}\right)e \\ -c(x) - \sigma\mu\Lambda^{-1}e \end{bmatrix}, \tag{5}$$

where $\Theta_L = (X - X_L)^{-1}N_L$, $\Theta_U = (X_U - X)^{-1}N_U$, and $D = \Lambda^{-1}S$. Note that if equality constraints are present in the problem, the diagonal matrix D includes some zero values in the rows corresponding to the equalities. The remaining variable and multiplier changes are computed by

$$\begin{aligned} \Delta s &= -s + \sigma\mu\Lambda^{-1}e - \Lambda^{-1}S\Delta\lambda \\ \Delta\nu_L &= -\nu_L + \sigma\mu(X - X_L)^{-1}e - (X - X_L)^{-1}N_L\Delta x \\ \Delta\nu_U &= -\nu_U + \sigma\mu(X_U - X)^{-1}e - (X_U - X)^{-1}N_U\Delta x \end{aligned} \tag{6}$$

using the Δx and $\Delta\lambda$ results of (5).

It is at this point that we exploit the structure of the Jacobian and Hessian matrices that result from problem (3). Because the local constraint groups depend only on global variables x_0 and their own respective local variables x_i, the Jacobian has the familiar *dual angular* block structure.[35]

$$J = \begin{bmatrix} J_{00} & & & \\ J_{10} & J_{11} & & \\ J_{20} & & J_{22} & \\ J_{30} & & & J_{33} \end{bmatrix} \tag{7}$$

Similarly, due to the objective function structure, the Hessian has a *bordered block diagonal* structure,[22]

$$H = \begin{bmatrix} H_{00} & H_{01} & H_{02} & H_{03} \\ H_{10} & H_{11} & & \\ H_{20} & & H_{22} & \\ H_{30} & & & H_{33} \end{bmatrix}. \tag{8}$$

Note that in MDO we generally do not have access to second derivative information due to the high computational cost incurred to obtain this information. When using a quasi-Newton method to estimate the Hessian, the approximate Hessian will generally be full. Nonetheless, near a local minimum, the terms shown in (8) can be expected to dominate in the approximate Hessian so our decomposition method will still be valid.

Because both the Hessian and Jacobian exhibit a sparse block structure, the linear system (5) is almost block separable. Ignoring the off-diagonal blocks in (8) and (7) would make (5) completely block-separable. Therefore, our first estimate to the solution of (5) uses block-diagonal estimates of J and H, denoted $\tilde{J}$ and $\tilde{H}$. Thus, the resulting linear system is

$$\begin{bmatrix} \tilde{H} + \Theta_L + \Theta_U & \tilde{J}^T \\ \tilde{J} & -D \end{bmatrix} \begin{bmatrix} \Delta\tilde{x} \\ \Delta\tilde{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla f - J^T\lambda - \sigma\mu\left((X_U - X)^{-1} - (X - X_L)^{-1}\right)e \\ -c(x) - \sigma\mu\Lambda^{-1}e \end{bmatrix}. \tag{9}$$

Depending on the number and magnitude of the coupling terms present in the off-diagonal (truncated) blocks, the solution estimate may or may not be sufficiently accurate for the overall algorithm to proceed. In this case, the accuracy of the solution is measured by taking the norm of the residual of (5) using $\Delta\tilde{x}$ and $\Delta\tilde{\lambda}$ computed from (9) and comparing it to the norm of the right-hand side of (5). If the estimated solution has not reduced the residual sufficiently, i.e., the residual (Euclidian) norm $||r|| \nleq \eta||b||$ for right-hand side norm $||b||$ and $0 \leq \eta < 1$, where η is chosen by the optimization algorithm, the solution can be refined using an appropriate Krylov method. This refinement step accelerates the convergence of the algorithm beyond that of a traditional decomposition algorithm because it allows the coupling terms to influence the search direction.

As with all Krylov methods, preconditioning may be used to reduce the number of refinement iterations. A straightforward choice of preconditioner, as pointed out by Conejo et al.,[23] is to use the inverse of the block diagonal system used to generate the initial estimate of the solution. In other words, use the inverse of the matrix in (9) as a preconditioner for (5). Unfortunately, this matrix is indefinite and not all Krylov methods accept indefinite preconditioners. A positive definite alternative can be derived from the Schur-complement of the indefinite system. Thus, the inverse of

$$\begin{bmatrix} \tilde{H} + \Theta_L + \Theta_U & 0 \\ 0 & D + \tilde{J}(\tilde{H} + \Theta_L + \Theta_U)^{-1}\tilde{J}^T \end{bmatrix} \tag{10}$$

may be used to precondition (5). In a problem containing equality constraints, the diagonal matrix D is not invertible. As a result, the other Schur-complement form containing D^{-1} cannot generally be formed. Note that both the indefinite and postive definite preconditioners are themselves block-separable matrices so they are particularly amenable to coarse-grained parallel computing. In addition, because Krylov methods do not require explicit storage of the matrix, the matrix structure in (5) can also be exploited in a coarse-grained parallel computing environment.

We call this approach to solving the linear system Block Approximation with Krylov Refinement, or BAKR. Within this general scheme, a number of Krylov methods and preconditioners may be chosen based on the observed matrix structure. The method used to select the η parameter at each iteration is also flexible, provided $\eta \to 0$ as optimality is approached.[31] Note also that we do not necessarily need the optimization problem to have a complicating variables structure for BAKR to be used. All we require are block-separable approximations of the Hessian and Jacobian matrices to obtain an initial solution estimate and a preconditioner. However, we expect the computational cost of the method to depend strongly on the choice of approximations.

Algorithm 1 describes our implementation of BAKR within an interior point algorithm. Note that, because interior point methods use a linearization of the KKT conditions, we may use the right hand side of (5) to measure optimality of the current point directly. A large family of interior point algorithms exists based on the multitude of parameter update schemes, global convergence strategies, convergence criteria, and other tools presented in the optimization literature. However, our principal contribution of the decomposed, inexact linear system solution should be compatible with any interior point method or even any sequential quadratic programming method.

Algorithm 1 A generic interior point algorithm with a BAKR procedure.

Select initial feasible point $x^{(0)}, \lambda^{(0)}, s^{(0)}, \nu_L^{(0)}, \nu_U^{(0)}$.
Select $\eta^{(0)}$ and convergence tolerance ϵ_{tol}.
Compute objective, constraints, and respective gradients.
while $||b^{(k)}|| > \epsilon_{tol}$ **do**
  Solve (9) to obtain an initial solution estimate.
  Compute the residual norm $||r||$ using the solution estimate in system (5)
  if $||r|| \leq \eta^{(k)}||b^{(k)}||$ **then**
    Keep solution estimate.
  else if $||r|| > \eta^{(k)}||b^{(k)}||$ and $||r|| \leq ||b^{(k)}||$ **then**
    while $||r|| > \eta^{(k)}||b^{(k)}||$ **do**
      Refine the solution estimate using one iteration of GMRES[36] with preconditioner (10).
    end while
  else
    Discard solution estimate; it is a bad approximation. Instead, use the zero vector as the starting point.
    while $||r|| > \eta^{(k)}||b^{(k)}||$ **do**
      Refine the solution estimate using one iteration of GMRES with preconditioner (10).
    end while
  end if
  Perform a backtracking line search using the computed search direction to improve the objective function, reduce constraint infeasibility, or both.
  Get new gradient information and update the estimated Hessian of the Lagrangian
  Update μ, σ, and any other algorithmic parameters.
  if The line search returns an improved point **then**
    $\eta^{(k+1)} \leftarrow \min\{\eta^{(k)}, 0.5||b^{(k+1)}||\}$
  else
    $\eta^{(k+1)} \leftarrow \max\{0.1\eta^{(k)}, 0.1\epsilon_{tol}\}$; The line search failure may be caused by a search direction that is insufficiently accurate.
  end if
  $k \leftarrow k + 1$
end while

IV. Results

In all of the following tests, the Hessian of the Lagrangian was computed using the BFGS quasi-Newton update formula. Global convergence was obtained using a backtracking line search with an l_1 merit function.[28] The complementarity parameter μ was computed at each iteration according to

$$\mu = \frac{\nu_L^T(x - x_L) + \nu_U^T(x_U - x) + \lambda^T s}{2n + m},$$

where n is the number of variables and m is the number of inequality constraints. The centering parameter σ was chosen adaptively using the same strategy as the LOQO code,[37] where

$$\sigma = 0.1 \min\left\{0.05\frac{1-\xi}{\xi}, 2\right\}^3, \text{ and } \xi = \mu^{-1}\min\{\nu_{Li}(x_i - x_{Li}), \nu_{Ui}(x_{Ui} - x_i), \lambda_j s_j\}.$$

The algorithm was deemed to be converged if the norm of the KKT residuals was less than 10^{-6} or if the norm of the step length was less than 10^{-12} and $\eta < 10^{-6}$. All problems were formed and solved using the Python programming language with the NumPy library providing vectorized array calculations, including linear system solution methods, and the SciPy library providing the GMRES implementation. Computations were performed using an Intel 2.53 GHz Core 2 Duo processor running Ubuntu Linux with 4GB of RAM. Note that since the computations were undertaken using sequential processing, further performance increases should be available across all problems once parallel processing is fully exploited.

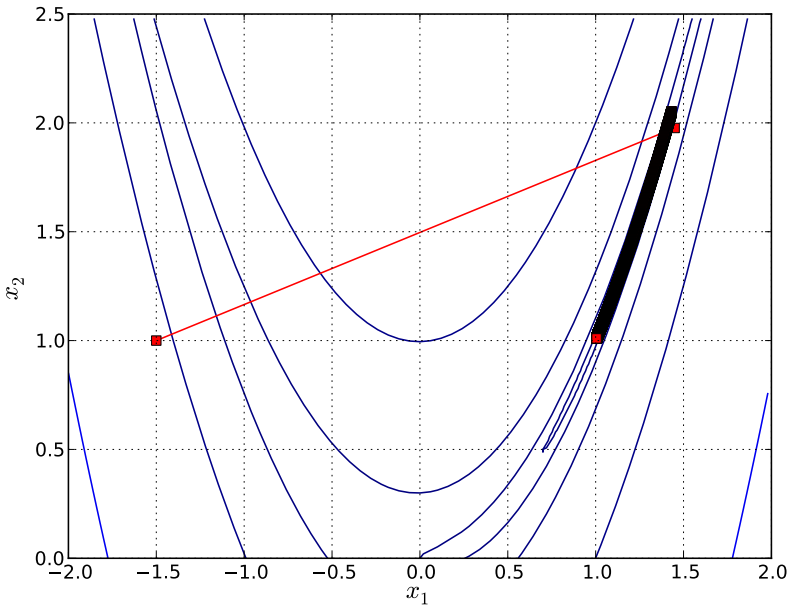

Figure 1. Convergence of the Rosenbrock problem without refining the decomposed solution.

The first test problem solved using BAKR with an interior point method is the Rosenbrock problem:

$$\begin{aligned} \text{minimize} \quad & f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \\ \text{with respect to} \quad & x_1, x_2 \end{aligned} \tag{11}$$

The optimal solution of this problem is at $(1, 1)^T$. The starting point used was $(-1.5, 1)^T$. This problem is used as an illustrative example to compare BAKR with a block-Newton strategy that truncates the coupling information.

Because of the strong coupling between x_1 and x_2 in the objective function, the block Newton approach (a type of coordinate descent algorithm) performs very poorly on this problem. However, because of the refinement step in BAKR, the curvature information can be used to refine the initial guess of the search direction, vastly improving convergence. Figures 1 and 2 compare the search paths without and with the refinement step from the same starting point. With refinement, only 19 iterations are required to solve the problem. Without refinement, the algorithm fails to converge even after 5000 iterations.

In order to see how BAKR can reduce computational cost, we need to solve a larger optimization problem where the effect becomes apparent. We also need to introduce nonlinear constraints to demonstrate the general applicability of the method. A problem that has both large size and a nonlinear constraint is a constrained multidimensional analogue to the Rosenbrock problem (11), i.e.,

$$\begin{aligned} \text{minimize} \quad & f(x) = \sum_{i=1}^{N-1} (100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & \sum_{i=1}^{N-1} (0.1 - (x_i - 1)^3 - (x_{i+1} - 1)) \leq 0 \\ & -5.12 \leq x \leq 5.12 \end{aligned} \tag{12}$$

An important property of this problem is that the exact Hessian of the objective function is tridiagonal so the problem is an attractive one for decomposition. For all trials, the initial point was $(4, ..., 4)^T$. The optimal

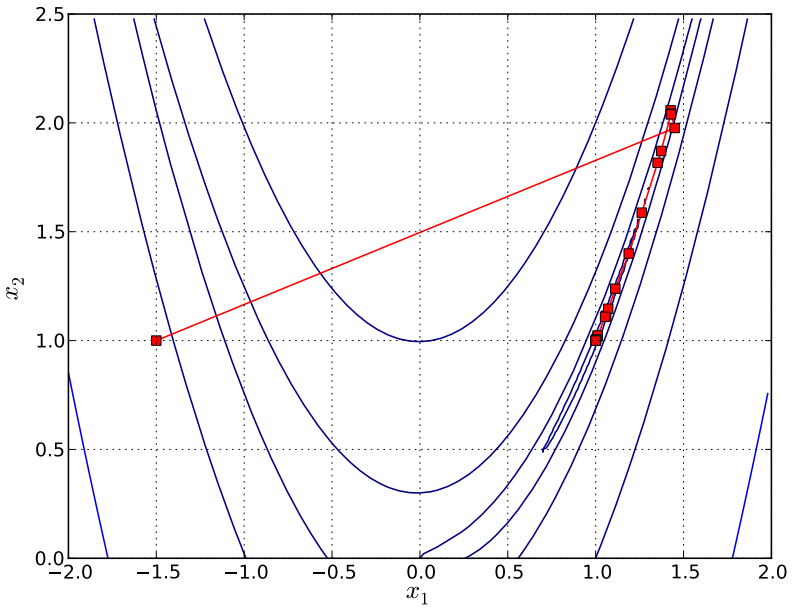

Figure 2. Convergence of the Rosenbrock problem with a BAKR procedure.

solution of the problem is near the unconstrained global minimum of $(1, ..., 1)^T$, but the exact solution depends on the problem dimension. The SQP optimizer SNOPT[38] was used to compute a numerically exact optimal solution for comparison with the results obtained from our own code.

Even though this problem does not fit the complicating variables model (3) in the form presented, it can still be solved using the BAKR approach. As discussed in Section III, the key elements needed for BAKR are block separable estimates of the Hessian and Jacobian matrices. In theory, the better these approximate matrices match the original matrices, the smaller the number of refinement steps needed by the Krylov method. For problem (12), the variable set is divided into five equally sized groups consisting of x_1 to $x_{N/5}$, $x_{N/5+1}$ to $x_{2N/5}$, etc., such that the number of significant coupling terms is as small as possible. The single coupling constraint is assumed to be influenced only by the last group of variables so that only the last $N/5$ columns of the approximate Jacobian are nonzero.

Figure 3 shows the total computational time with the number of design variables ranging between one hundred and one thousand. We compare three methods of solving the linear system:

1. Formation and direct factorization of the linear system to obtain an exact step. This method is called Direct Factorization in the figures.

2. Iterative solution of the linear system using GMRES but with no preconditioning or initial solution estimate. The solution tolerance η at each iteration is chosen in exactly the same way as in the BAKR method. This method is called GMRES in the figures.

3. The BAKR method outlined in Algorithm 1, referred to as BAKR in the figures.

In all respects other than the linear system solve, the optimization algorithms are identical. The results indicate that while solution times are very close for all methods when the number of variables is small, BAKR is effective at reducing computational time when the number of variables is large. The performance of both the direct and iterative matrix solves is similar across a wide range of problem sizes. For the thousand-variable case, the solution time for BAKR is about one third of that required by either direct or iterative matrix methods. The average slopes of the trendlines indicate that, for an order of magnitude increase in problem size, the BAKR method reduces computational cost by half an order of magnitude compared with the other methods.

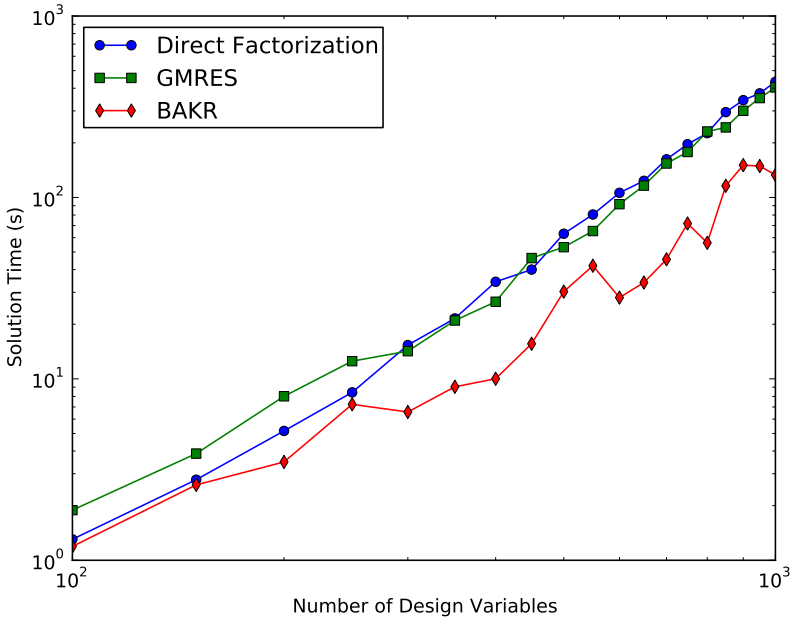

Figure 3. Computational times needed to solve problem (12)**.**

As a check on the accuracy of our code, we computed the distance (Euclidian norm of the difference) between the KKT point found by our code and the KKT point found by SNOPT. Figure 4 plots this data as a function of problem size. While robustness of our code is an issue, as evidenced by the behavior of the data in Figure 4, observe that our optimizer still finds the minimum regularly when the BAKR method is used. Thus, use of BAKR as a linear solver does not compromise the robustness of the overall optimization algorithm.

While the results for the solution time appear promising, a more representative measure of performance on MDO problems is the number of objective and constraint function evaluations required by the method. This metric serves as a proxy for the number of expensive discipline analysis calls required in a realistic design problem. Because the disciplinary evaluations tend to dominate the computational time in the solution of an MDO problem, the number of function calls is, therefore, an indirect estimate of solution time in an MDO problem.

Figure 5 compares the number of function calls for all three matrix methods. As with solution time, the performace of direct and iterative solve methods is similar, while BAKR shows substantially improved performance. On average, over the range of problem sizes tested, the optimizer using BAKR requires about half the number of function evaluations compared to the optimizers that do not employ BAKR.

The final result to check for this problem is the effect of the preconditioner and initial solution estimate on the number of Krylov iterations. Figure 6 compares the number of GMRES iterations required by the two iterative solve approaches as the number of variables increases. In most cases, BAKR requires an order of magnitude fewer GMRES iterations than the unpreconditioned iterative solve. This is a strong demonstration of the effectiveness of the approximate Schur preconditioner (10).

Another scalable test problem is the scalable quadratic MDO problem devised by Tedford and Martins,[9]

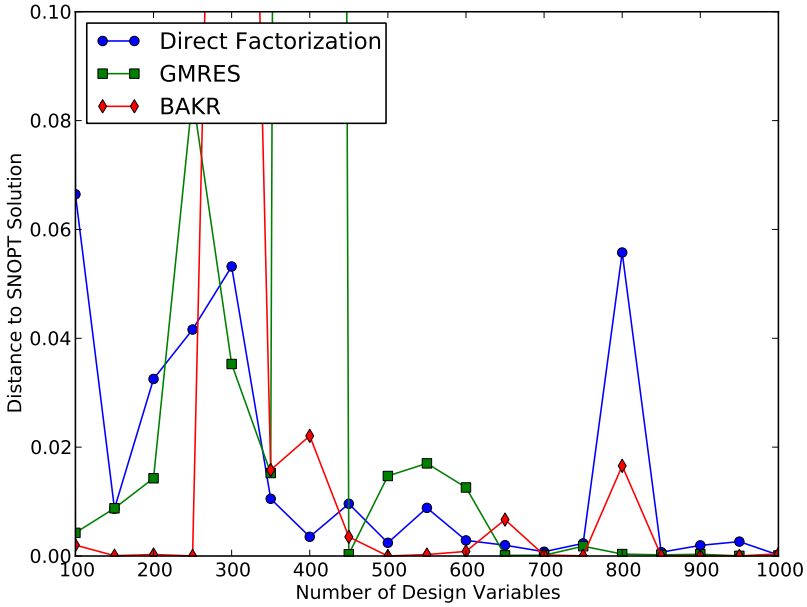

Figure 4. Distance from trial solutions to the exact solutions determined by SNOPT for problem (12).

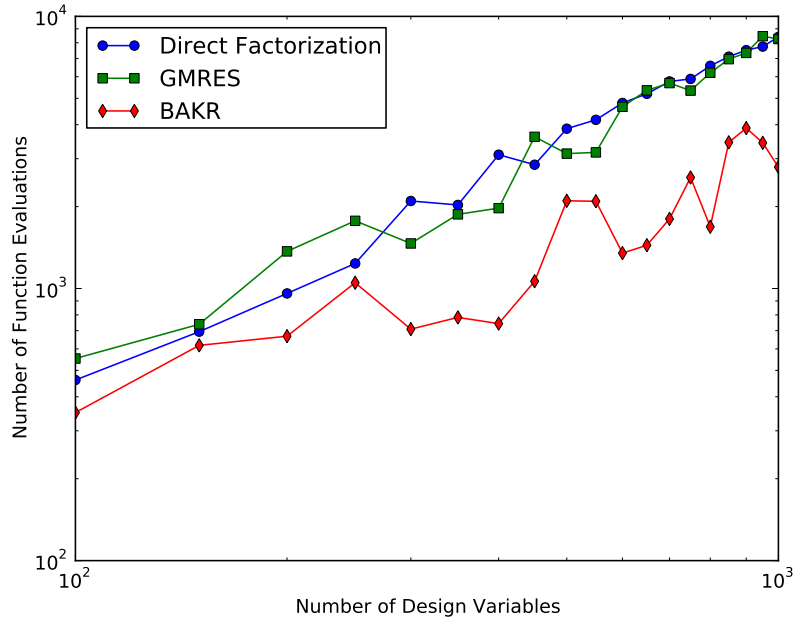

Figure 5. Function calls needed to solve problem (12).

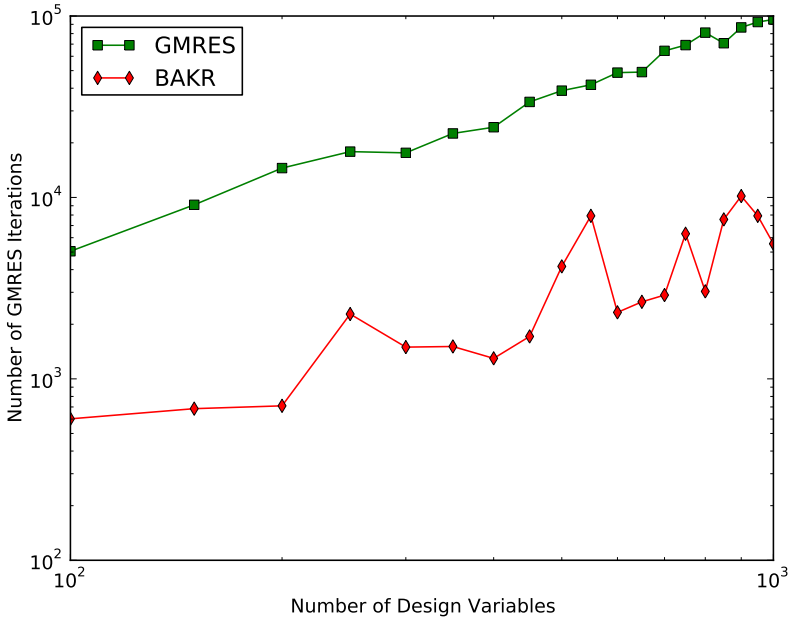

Figure 6. Number of GMRES iterations in iterative solve methods for problem (12)**.**

which in the IDF formulation is

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{n_0} x_{0k}^2 + \sum_{i=1}^{N} \sum_{l=1}^{n_y} (y_{il}^t)^2 \\ \text{with respect to} \quad & x, y^t \\ \text{subject to} \quad & 1 - \frac{y_i^t}{C_i} \leq 0 \\ & y_i^t - y_i(x_0, x_i, y_{j \neq i}^t) = 0 \\ \text{where} \quad & C_{y_i} y_i = - \left(C_{x_{0i}} x_{0i} + C_{x_i} x_i - \sum_{j=1, j \neq i}^{N} C_{y_j} y_j^t \right), \end{aligned} \tag{13}$$

where each constraint and discipline group applies for $i = 1, ..., N$. The C_i, C_{x_i}, C_{y_i}, and C_{z_i} terms are matrices of appropriate size composed of random coefficients between zero and ten. In addition, every other C_{y_i} matrix is changed from positive to negative. In the current implementation, we use only three disciplines, ten shared design variables, and 50 coupling variables from each discipline with a varying number of local design variables. The initial point for the optimization was obtained by choosing all design variables to be equal to one and performing a multidisciplinary analysis to obtain initial coupling targets. Using all three matrix solve methods, the minimum point of this problem was found in all cases. As with problem (12), verification was accomplished by running SNOPT on these problems and comparing the solutions.

Unfortunately, results for this problem are disappointing. Figures 7 and 8 show the solution time and number of function calls required for problem (13) with an increasing number of local design variables. Clearly, compared to the direct and iterative matrix methods, BAKR shows no significant advantage in solution time or the number of function calls. The reasons for this apparent failure are still under investigation. One area of concern is the problem structure itself. Because problem (13) contains a quadratic objective function with linear constraints, only a few interior point optimizer iterations are required to obtain convergence regardless of the linear system solver employed. Thus, the large linear systems are not solved enough times to obtain a reliable estimate of the relative performance of each method.

The positive result from problem (13) lies in the performance of the preconditioner within BAKR. Figure 9

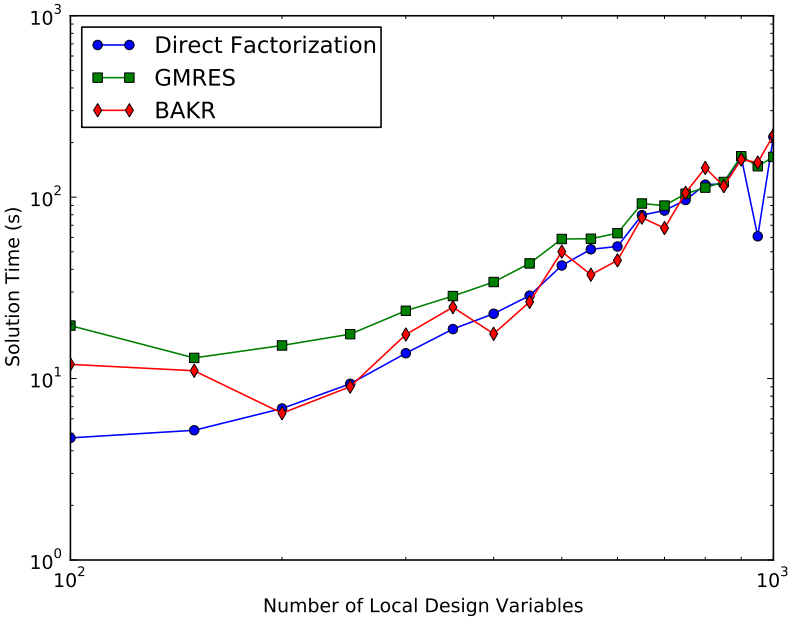

Figure 7. Computational times needed to solve problem (13).

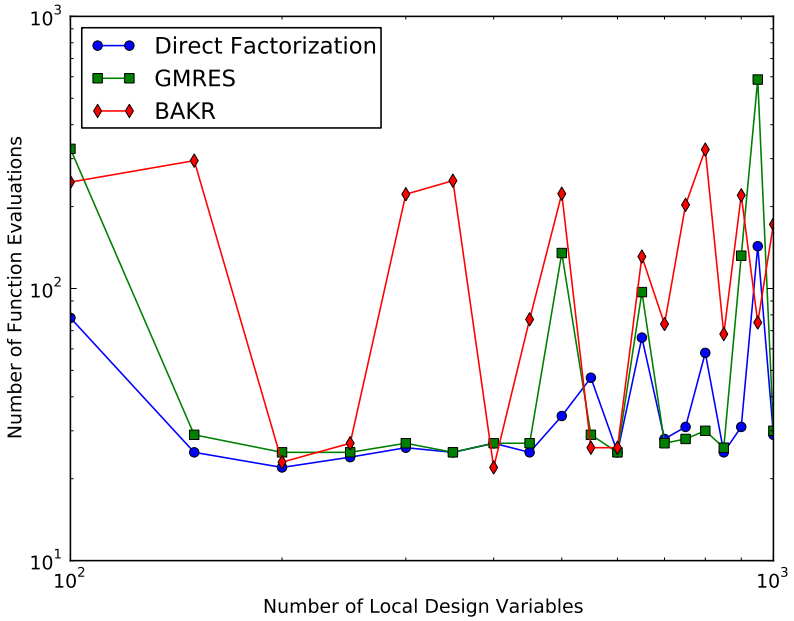

Figure 8. Function calls needed to solve problem (13).

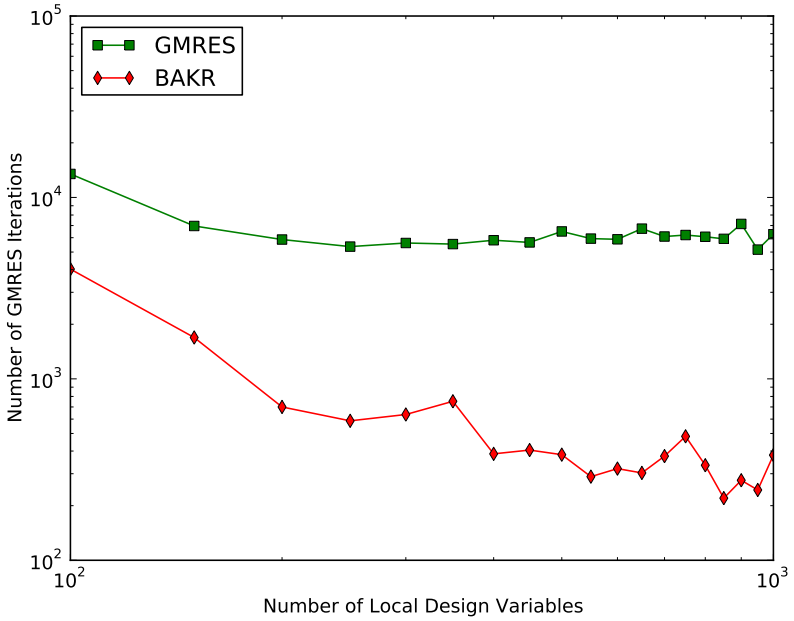

Figure 9. Number of GMRES iterations in iterative solve methods for problem (13)**.**

compares the number of GMRES iterations required by BAKR and the generic iterative solver with an increasing number of local design variables. As with Figure 6, compared to the generic iterative solver, BAKR reduces number of GMRES iterations by an order of magnitude or better. This result gives further confirmation of the effectiveness of preconditioner (10).

V. Conclusion

In this work, we presented a new approach to the decomposition of MDO problems we call Block Approximation with Krylov Refinement, or BAKR. This new method is a type of internal decomposition in that it is implemented *within* an optimization algorithm rather than outside of one. The basic idea is to estimate the solution of the large linear systems generated by an SQP or interior point algorithm using approximate systems with block-separable estimates of the Hessian and Jacobian matrices. The solution estimate is then refined to the desired level of accuracy using a preconditioned Krylov method applied to the original linear system. Convergence of the optimization algorithm using these inexact steps is guaranteed by the theory of inexact Newton and quasi-Newton methods.

Preliminary results from solving scalable test problems via an interior point method are encouraging. They show that if the structure of large optimization problems is exploited, significant reductions in computational time and function evaluations are possible. However, more testing is needed to confirm this trend. The observed performance gains are evident even when parallel processing is not employed to the fullest extent. We emphasize that this improvement is due to the specialized algorithm used to solve the linear system, and no other property of overall optimization method.

Future work in this area includes additional testing on larger, more realistic engineering problems to confirm these preliminary results. We also wish to incorporate coarse-grained parallel computing on these problems to test parallel efficiency. Areas of improvement to the existing optimization algorithm include further exploiting MDO problem structure by using specialized quasi-Newton methods to account for the block structure of the Hessian and improving the line search procedure to further reduce the number of function calls. In addition, the symmetry of the linear systems under consideration has not been exploited and using a Krylov method specialized for a symmetric linear system should yield an additional improvement in method robustness and efficiency.

Acknowledgments

Funding for this work was provided by the Natural Science and Engineering Research Council of Canada and the University of Toronto.

References

[1]Haftka, R. T., Sobieszczanski-Sobieski, J., and Padula, S. L., "On Options for Interdisciplinary Analysis and Design Optimization," *Structural Optimization*, Vol. 4, 1992, pp. 65–74.

[2]Sobieszczanski-Sobieski, J. and Haftka, R. T., "Multidisciplinary aerospace design optimization: survey of recent developments," *Structural Optimization*, Vol. 14, No. 1, 1997, pp. 1–23.

[3]Kroo, I. M., "MDO for Large-Scale Design," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. Alexandrov and M. Y. Hussaini, SIAM, 1997, pp. 22–44.

[4]Alexandrov, N. M. and Lewis, R. M., "Analytical and Computational Aspects of Collaborative Optimization for Multidisciplinary Design," *AIAA Journal*, Vol. 40, No. 2, 2002, pp. 301–309.

[5]Chittick, I. R. and Martins, J. R. R. A., "Aero-Structural Optimization Using Adjoint Coupled Post-Optimality Sensitivities," *Structures and Multidisciplinary Optimization*, Vol. 36, 2008, pp. 59–70.

[6]DeMiguel, V. and Nogales, F. J., "On Decomposition Methods for a Class of Partially Separable Nonlinear Programs," *Mathematics of Operations Research*, Vol. 33, No. 1, February 2008, pp. 119–139.

[7]Martins, J. R. R. A., Marriage, C., and Tedford, N., "pyMDO: An Object-Oriented Framework for Multidisciplinary Design Optimization," *ACM Transactions on Mathematical Software*, Vol. 36, No. 4, 2009, pp. 2–25.

[8]Perez, R. E., Liu, H. H. T., and Behdinan, K., "Evaluation of Multidisciplinary Optimization Approaches for Aircraft Conceptual Design," *Proceedings of the 10th AIAA/ISSMO Muldisiciplinary Analysis and Optimization Conference*, Albany, NY, Aug. 2004, AIAA 2004-4537.

[9]Tedford, N. P. and Martins, J. R. R. A., "Benchmarking Multidisciplinary Design Optimization Algorithms," *Optimization and Engineering*, Vol. 11, 2010, pp. 159–183.

[10]Yi, S. I., Shin, J. K., and Park, G. J., "Comparison of MDO Methods with Mathematical Examples," *Structural and Multidisciplinary Optimization*, Vol. 39, 2008, pp. 391–402.

[11]Braun, R. D., Gage, P., Kroo, I. M., and Sobieski, I. P., "Implementation and Performance Issues in Collaborative Optimization," *Proceedings of the* 6^{th} *AIAA, NASA, and ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1996, AIAA Paper 1996-4017.

[12]Sobieszczanski-Sobieski, J., Altus, T. D., Phillips, M., and Sandusky, R. R., "Bi-Level Integrated System Synthesis for Concurrent and Distributed Processing," *AIAA Journal*, Vol. 41, No. 10, October 2003, pp. 1996–2003.

[13]Roth, B. and Kroo, I., "Enhanced Collaborative Optimization: Application to an Analytic Test Problem and Aircraft Design," *Proceedings of the* 12^{th} *AIAA/ISSMO Multidisciplinary Analysis and Optimizaton Conference*, Victoria, British Columbia, Canada, September 2008, AIAA 2008-5841.

[14]Kim, H. M., Michelena, N. F., Papalambros, P. Y., and Jian, T., "Target Cascading in Optimal System Design," *Journal of Mechanical Design*, Vol. 125, No. 3, September 2003, pp. 474–480.

[15]Michalek, J. J. and Papalambros, P. Y., "An Efficient Weighting Update Method to Achieve Acceptable Consistency Deviation in Analytical Target Cascading," *Journal of Mechanical Design*, Vol. 127, March 2005, pp. 206–214.

[16]Kim, H. M., Chen, W., and Wiecek, M. M., "Lagrangian Coordination for Enhancing the Convergence of Analytical Target Cascading," *AIAA Journal*, Vol. 44, No. 10, October 2006, pp. 2197–2207.

[17]Tosserams, S., Etman, L. F. P., and Rooda, J. E., "An Augmented Lagrangian Decomposition Method for Quasiseparable Problems in MDO," *Structural and Multidisciplinary Optimization*, Vol. 34, 2007, pp. 211–227.

[18]Tosserams, S., Etman, L. F. P., and Rooda, J. E., "Augmented Lagrangian Coordination for Distributed Optimal Design in MDO," *International Journal for Numerical Methods in Engineering*, Vol. 73, 2008, pp. 1885–1910.

[19]Laird, C. D. and Biegler, L. T., "Large-Scale Nonlinear Programming for Multi-scenario Optimization," *Modeling, Simulation, and Optimization of Complex Processes*, edited by H. G. Bock, E. Kostina, H. X. Phu, and R. Rannacher, Springer-Verlag, Berlin, 2008.

[20]Zavala, V. M., Laird, C. D., and Biegler, L. T., "Interior-point Decomposition Approaches for Parallel Solution of Large-Scale Nonlinear Parameter Estimation Problems," *Chemical Engineering Science*, Vol. 63, No. 19, October 2008, pp. 4834–4845.

[21]Gondzio, J. and Grothey, A., "Direct Solution of Linear Systems of Size 10^9 Arising in Optimization with Interior Point Methods," *Parallel Processing and Applied Mathematics 2005*, edited by R. Wyrzykowski, J. Dongarra, N. Meyer, and J. Wasniewski, Vol. 3911 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2006, pp. 513–525.

[22]Gondzio, J. and Grothey, A., "Exploiting Structure in Parallel Implementation of Interior Point Methods for Optimization," *Computational Management Science*, Vol. 6, No. 2, 2009, pp. 135–160.

[23]Conejo, A. J., Nogales, F. J., and Prieto, F. J., "A Decomposition Procedure Based on Approximate Newton Directions," *Mathematical Programming*, Vol. 493, 2002, pp. 495–515.

[24]Conejo, A. J., Castillo, E., Minguez, R., and Garcia-Bertrand, R., *Decomposition Techniques in Mathematical Programming: Engineering and Science Applications*, Springer-Verlag, 2006.

[25]Cohen, G., "Optimization by Decomposition and Coordination: A Unified Approach," *IEEE Transactions on Automatic Control*, Vol. AC–23, No. 2, April 1978, pp. 222–232.

[26]Cohen, G., "Auxiliary Problem Principle and Decomposition of Optimization Problems," *Journal of Optimization Theory and Applications*, Vol. 32, No. 3, November 1980, pp. 277–305.

[27]Patriksson, M., "Decomposition Methods for Differentiable Optimization over Cartesian Product Sets," *Computational Optimization and Applications*, Vol. 9, 1998, pp. 5–42.

[28]Nocedal, J. and Wright, S. J., *Numerical Optimization*, Springer-Verlag, 2nd ed., 2006.

[29]Cramer, E. J., Dennis, J. E., Frank, P. D., Lewis, R. M., and Shubin, G. R., "Problem Formulation for Multidisciplinary Optimization," *SIAM Journal on Optimization*, Vol. 4, No. 4, 1994, pp. 754–776.

[30]Haftka, R. T. and Watson, L. T., "Multidisciplinary Design Optimization with Quasiseparable Subsystems," *Optimization and Engineering*, Vol. 6, 2005, pp. 9–20.

[31]Dembo, R. S., Eisenstat, S. C., and Steihaug, T., "Inexact Newton Methods," *SIAM Journal on Numerical Analysis*, Vol. 19, No. 2, April 1982, pp. 400–408.

[32]Steihaug, T., "Local and Superlinear Convergence for Truncated Iterated Projection Methods," *Mathematical Programming*, Vol. 27, 1983, pp. 176–190.

[33]Wright, S. J., *Primal-Dual Interior Point Methods*, SIAM, 1997.

[34]Wächter, A. and Biegler, L. T., "On the Implementation of an Interior Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, Vol. 106, 2006, pp. 25–57.

[35]Lasdon, L. S., *Optimization Theory for Large Systems*, The Macmillan Company, 1970.

[36]Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific Computing*, Vol. 7, 1986, pp. 856–869.

[37]Vanderbei, R. J., "LOQO: An Interior Point Code for Quadratic Programming," *Optimization Methods and Software*, Vol. 11, 1999, pp. 451–484.

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