Optim Eng (2010) 11: 159–183
DOI 10.1007/s11081-009-9082-6

Benchmarking multidisciplinary design optimization algorithms

Nathan P. Tedford · Joaquim R.R.A. Martins

Received: 28 March 2007 / Accepted: 26 February 2009 / Published online: 20 March 2009

Abstract A comparison of algorithms for multidisciplinary design optimization (MDO) is performed with the aid of a new software framework. This framework, pyMDO, was developed in Python and is shown to be an excellent platform for comparing the performance of the various MDO methods. pyMDO eliminates the need for reformulation when solving a given problem using different MDO methods: once a problem has been described, it can automatically be cast into any method. In addition, the modular design of pyMDO allows rapid development and benchmarking of new methods. Results generated from this study provide a strong foundation for identifying the performance trends of various methods with several types of problems.

Keywords Multidisciplinary design optimization · Decomposition algorithms · Nonlinear programming · Sensitivity analysis

1 Introduction

Multidisciplinary design optimization (MDO) is a growing field of research with a wide range of applications. When optimizing engineering systems that involve multiple disciplines, it is well known that strategies such as sequential optimization are often not able to find the true optimum of the system. Thus it is important that interdisciplinary interactions be accounted for properly, both in the simulation of the coupled systems and the optimization. Only by considering these interactions during the optimization process can the true optimum of the coupled system be determined.

As research on MDO has matured, the number of methods available to solve a given problem has increased. These methods can be divided into two classes: monolithic formulations and multilevel formulations. Monolithic formulations, which include the multidisciplinary design feasible (MDF) and the simultaneous analysis and

N.P. Tedford · J.R.R.A. Martins (✉)
University of Toronto Institute for Aerospace Studies, Toronto, ON M3H 5T6, Canada
e-mail: martins@utias.utoronto.ca

design (SAND) approaches, use a single system-level optimizer for the whole problem. These approaches tend to be the most straightforward to implement for small problems, but can scale poorly with large problems and many disciplines. Additionally, the structure of these formulations is such that they may not adapt well to an industrial design setting, where groups in charge of each discipline may work largely independently of one another. Multilevel methods such as collaborative optimization (CO) (Braun and Kroo 1997), concurrent subspace optimization (CSSO) (Wujek et al. 1997), and bilevel integrated systems synthesis (BLISS) (Sobieszczanski-Sobieski et al. 2003) use subspace optimizations to promote discipline autonomy. The system-level optimizer is then responsible for managing the interactions between the discipline optimizations. This approach mimics an industrial setting more closely and allows each disciplinary subgroup to work in relative isolation based on design targets provided by the system-level optimizer.

With the various MDO methods available, how does one decide which one to use for a given MDO problem? Typically the selection of an MDO method is done in an *ad hoc* manner, since few benchmarking studies are available to make an informed decision (Alexandrov and Kodiyalam 1998). Results from various studies have shown that the performance of a method can be dependent on its implementation, the characteristics of the problem being solved, and the optimizer employed (Alexandrov and Kodiyalam 1998; Brown and Olds 2006; Perez et al. 2004). Furthermore, for some problems, specific methods may either fail to return an optimum, or may not be suited to implementation (Brown and Olds 2006). Additionally, comparing results between studies can be difficult as the performance of an MDO method can depend on specific implementation details.

Furthermore, after selecting an MDO method, it can be difficult to determine its proper or most efficient implementation. In the case of collaborative optimization, there are at least four major variants (Alexandrov and Lewis 2002; Sobieski and Kroo 2000; Braun et al. 1996; Braun and Kroo 1997). Though many of the implementations have been used to solve specific problems, there has been no study that thoroughly tested each implementation in a statistically significant manner. Therefore, it is left to the practitioner to search through the literature in an attempt to find an implementation that may work well for their particular problem.

pyMDO—the software used in this study—addresses a number of issues (Martins et al. 2008). The main goal is to create a framework that allows the testing of many MDO problems in an efficient manner to facilitate the benchmarking of MDO methods and the development of new ones. By removing the need to manually implement the various MDO methods for each problem, comparisons using pyMDO can focus on quantitative metrics such as the number of function evaluations, optimal variable accuracy, and convergence rate.

In this paper, the computational performance of MDF, IDF, SAND, CO, and CSSO are compared by solving a suite of four MDO problems. BLISS is not benchmarked in this paper, as it requires coupled sensitivities. The automatic implementation of this computation is rather involved and is the subject of ongoing research. The effect of the sensitivity accuracies on the performance of each MDO method is also investigated. The description of the each MDO method is followed by an overview of the pyMDO framework. We then present the problem suite and the corresponding benchmarking results.

2 Multidisciplinary systems

A defining characteristic of MDO problems is the existence of two or more design disciplines. Each discipline is given design authority over part of the system and is responsible for solving a set of governing equations to find the state variables given an appropriate set of inputs. Disciplines require not only design variables, but also information related to the state of other disciplines in the problem. This relationship between the inputs of one discipline and the outputs of another results in a coupled system.

Traditionally, the coupled multidisciplinary system has been solved through the use of block-iterative methods. Each discipline is provided with a set of design variables and disciplinary inputs (non-local coupling variables) and is responsible for generating a set of discipline feasible states and outputs (local coupling variables) by solving the required governing equations. The solution of the coupled system can be represented as,

$$\mathcal{R}_i(z, x_i, y_j) = 0, \quad i = 1, \ldots, N, \tag{1}$$

where i denotes the discipline, $j \neq i$ and N is the total number of disciplines in the problem. In MDO problems, the design variables can be subdivided based on their effect on various disciplines. Design variables required to solve only the governing equations of one discipline are considered *local* to that discipline. Design variables local to discipline i are denoted by x_i in (1). Design variables that affect two or more disciplines are considered *global*. These are denoted by z in the above equation.

Each discipline i generates feasible states (y_i) by solving a series of governing equations ($\mathcal{R}_i$) given an appropriate set of inputs (z, x_i, y_j). These governing equations (1) can be seen as equality constraints.

The set of discipline governing equations are solved in succession until the change in the coupling variable sets over successive iterations is within a specified tolerance. This multidisciplinary convergence criterion can be stated as,

$$y_i^{m+1} = y_i^m, \quad i = 1, \ldots, N, \tag{2}$$

where y_i^m represents the value of discipline i's coupling variables after m iterations.

MDO methods are defined by how they transform a multidisciplinary problem into one or a series of optimization problems. We will now describe these methods.

3 MDO architectures

MDO methods transform the multidisciplinary problem into either a single problem or a series of optimization problems that can be solved through the use of standard numerical optimization algorithms. There are a wide variety of different methods, each of which formulates the given problem in a different manner. Each method has its advantages and disadvantages and their performance is very much problem dependent. A variety of metrics can be used to classify MDO problems. Some of these include the number of design variables, the ratio between global and local design variables, the number of coupling variables, and the number of disciplines. This list

is by no means exhaustive, but covers only some of the more important aspects of MDO problems that must be examined prior to the selection of an MDO method. In this section we describe the various methods.

3.1 Multidisciplinary feasible

The multidisciplinary feasible (MDF) method is the traditional MDO approach and it involves solving a single optimization problem that calls a multidisciplinary analysis (MDA) when objective or constraint values are required. The MDA solves the governing equations for all disciplines. The MDA module takes the design variables solves all governing equations until the coupling variables have converged. The values of the objective and constraints can then computed (Cramer et al. 1994).

By requiring the solution of the MDA at each design point, MDF ensures that each optimization iteration is multidisciplinary feasible. This is a very desirable property, since if the optimization is terminated prematurely, a physically realizable design point is at hand.

The effort required to implement MDF for a given problem is directly related to the effort required to build an appropriate MDA module. General iterative approaches, such as block Gauss–Seidel iteration, can be used in most situations, but do not always converge. The use of under-relaxation can help to improve the convergence properties, but it is not always effective. Another shortcoming of block-iterative schemes is that they are much less efficient than fully integrated MDA solvers. For example, in a problem where all the discipline analyses are linear, Newton solvers are computationally much more efficient than other iterative solvers.

Depending on the sensitivity analysis method employed, the use of gradient-based optimizers with MDF can result in poor performance. If either a finite difference or complex-step method (Martins et al. 2003) is used to calculate the sensitivities of the objective and constraints with respect to the design variables, an MDA must be solved for each component of the sensitivity. Since generating solutions to the MDA can be costly, performing even one complete sensitivity analysis can be prohibitively expensive with these methods. Semi-analytic sensitivity analysis methods should therefore be used whenever possible in order to reduce the total number of calls to the MDA module (Martins et al. 2005).

Another drawback of MDF is that there is little opportunity for parallel computation outside the MDA module. This can increase the computational cost of the method when compared to decoupled approaches that take advantage of parallel computations.

MDF can be mathematically stated as,

$$\begin{aligned} &\text{minimize:} && f\left(z, x, y\left(x, z\right)\right) \\ &\text{w.r.t.:} && z, x \\ &\text{s.t.:} && c\left(z, x, y\left(x, z\right)\right) \leq 0, \end{aligned} \tag{3}$$

where x represents the complete set of local design variables and y all the discipline coupling variables determined by the solution of the MDA. The constraints, c, can also be divided into local and global constraints.

Note that in the mathematical description of the methods, the dependence of the coupling variables of discipline i (y_i) on the its own state variables (u_i) is omitted. In general, coupling variables for discipline i depend on the design variables z and x_i as well as the state variables u_i. The state variables are obtained from the solution of the discipline analysis represented by $\mathcal{R}_i(z, x, y_j) = 0$, where y_j are the non-local coupling variables. Therefore, in the most general case the calculation of a the coupling variables for discipline i should be written as $y_i = y_i(z, x, y_j, u_i(z, x, y_j))$.

3.2 Individual discipline feasible

The individual discipline feasible (IDF) method (Cramer et al. 1994) is a decoupled version of MDF. Instead of enforcing multidisciplinary feasibility at each design point, IDF only enforces discipline feasibility. That is, at each optimization iteration, the governing equations (1) for each discipline are satisfied, but a multidisciplinary feasible solution may not have been realized. Therefore, if the optimization fails or is stopped prematurely, each discipline will have a realizable design, but the system design as a whole will likely be infeasible as the coupling variables (y) may not have converged according to the criterion (2).

To decouple the discipline analyses so that they no longer rely on one another for their coupling variable inputs, IDF adds the coupling variables to the set of design variables. The optimizer then provides each discipline with both design variables and an estimate of the coupling variables. To ensure that a multidisciplinary feasible solution is achieved at the optimum, one additional feasibility constraint is added to the optimization problem for each coupling variable. These constraints ensure that at the optimum, the estimate of the coupling variables matches the actual coupling variables computed by each discipline.

IDF can be stated as,

$$\begin{aligned} &\text{minimize:} && f(z, x, y^t) \\ &\text{w.r.t.:} && z, x, y^t \\ &\text{s.t.:} && c(z, x, y(x, y^t, z)) \leq 0 \\ & && y_i^t - y_i\left(x, y_j^t, z\right) = 0, \end{aligned} \tag{4}$$

where y^t represents the coupling variables estimates (or targets) provided by the optimizer, y_i are the coupling variable outputs of discipline i given the estimate of the non-local coupling variables y_j^t.

For a given set of design variables each discipline finds their respective local states by solving their governing equations, $\mathcal{R}_i(z, x, y_j^t) = 0$. Since these discipline governing equations are now uncoupled they can be solved in parallel, which can greatly increase the overall performance of the method. Additionally, the uncoupled nature of IDF requires that each discipline analysis be solved only once per design point. As discipline evaluations can be time consuming, an objective evaluation in IDF is much less costly than an evaluation in MDF. IDF is also more robust than methods involving an MDA, as obtaining an MDA solution can sometimes be problematic.

As with MDF, if a gradient-based optimizer is employed it is recommended that semi-analytic sensitivity analysis methods be used whenever possible. Since IDF increases both the number of design variables and the number of constraints, the cost of

sensitivity analysis per design point can very large if finite differences or the complex-step method are used. Unfortunately, no efficient semi-analytic method exists that efficiently computes the sensitivities of a large number of functions of interest with respect to an equally large number of variables. Thus for problems with a large number of both coupling variables and constraints, the cost of performing the sensitivity analysis needed for IDF can be prohibitive.

Therefore, it is preferable to use IDF with problems that exhibit a low bandwidth of coupling in order minimize the number of coupling variables and constraints that must be added to the optimization. Techniques such as aggregation can also be used in some problems to reduce the total number of coupling variables.

3.3 Simultaneous analysis and design

Simultaneous analysis and design (SAND) decomposes the multidisciplinary problem a step further than IDF by relaxing the requirement of discipline feasibility at each optimization iteration (Cramer et al. 1994). In this case, the optimizer is assigned with the task of solving the governing equations and the optimization problem simultaneously.

This is accomplished by treating the residuals of the discipline analyses (1) as equality constraints in the optimization problem. Conventional nonlinear programming experience suggests that optimal solutions can be found at a lower expense than other MDO methods. It should be noted that for small problems where the state variable set is identical to the coupling variable set, SAND is equivalent to IDF.

SAND can be stated as,

$$\begin{aligned} &\text{minimize:} && f(z, x, y(z, x, u)) \\ &\text{w.r.t.:} && z, x, u \\ &\text{s.t.:} && c(z, x, y(z, x, u)) \leq 0 \\ & && \mathcal{R}(z, x, y(z, x, u), u) = 0, \end{aligned} \tag{5}$$

where $\mathcal{R}(z, x, y(z, x, u), u)$ represents the residuals of the governing equations for all disciplines.

In this method, the state variables (u) for each discipline are added to the set of design variables. At each iteration, the state variables are used to compute the coupling variables necessary to evaluate the objective and the constrains. The residuals of each discipline are then evaluated to provide the values for the residual constraints.

Since the governing equations are not necessarily satisfied until the optimization problem is solved, discipline feasibility is not generally attained at intermediate design points. Therefore, if the optimization interrupted, the resulting design might not be physically realizable. Note that SAND does away with the governing equations solver for each discipline, but needs a function that computes the residuals of the governing equations. This is not always possible for compiled proprietary software and SAND cannot be implemented when that is the case.

The SAND method can drastically increase the dimensionality of the optimization problem. The cost of the sensitivity analyses required for gradient-based optimization can be prohibitive because the sensitivity matrix of residuals with respect to states

is square. As with IDF, even with an inexpensive analysis (in this case a residual evaluation) the calculation of sensitivities for problems with a large number of state variables can become prohibitive.

3.4 Collaborative optimization

Collaborative optimization (CO) was first proposed by Braun and Kroo (1997) and is the first of the bilevel methods described herein. CO is designed to provide discipline autonomy, while enforcing interdisciplinary compatibility. The optimization problem is decomposed into a number of independent optimization subproblems, each corresponding to one discipline. Each discipline optimization is given control over its local design variables and is responsible for satisfying its local constraints. Discipline feasibility is maintained throughout the system-level optimization process since the discipline optimizations are responsible for generating discipline feasible solutions for each system-level iteration. The introduction of compatibility constraints at the system level ensures that multidisciplinary feasibility is achieved when the system-level optimization problem has converged.

The CO system-level problem can be written as,

$$\begin{aligned} &\text{minimize:} && f(z, y, x_{\text{obj}}) \\ &\text{w.r.t.:} && z, y, x_{\text{obj}} \\ &\text{s.t.:} && J_i^* = 0, \quad i = 1, \ldots, N \end{aligned} \tag{6}$$

where J_i^* is a measure of the interdisciplinary compatibility of each discipline and is given by the solution of the following optimization problem for each discipline i,

$$\begin{aligned} &\text{minimize:} && J_i = \sum (z_i - z_i^t)^2 + \sum (x_{i_{\text{obj}}} - x_{i_{\text{obj}}}^t)^2 \\ & && \quad + \sum (y_i - y_i^t)^2 + \sum (y_{j_i} - y_{j_i}^t)^2 \\ &\text{w.r.t.:} && z_i, x_i, y_{j_i} \\ &\text{s.t.:} && c(x_i, z_i, y_i(x_i, y_{j_i}, z_i)) \leq 0 \end{aligned} \tag{7}$$

where superscript t represents system-level target values that are held constant throughout the discipline optimizations.

The design variables at the system level consist of global design variables, coupling variables, and any local design variables that explicitly affect the objective (x_{obj}). The system-level constraints consist of the global constraints and one compatibility constraint per discipline. There are a few variants of CO and each uses a slightly different form of the compatibility constraints. In the present framework, the CO_2 variant proposed by Alexandrov and Lewis (1999) is used with the following compatibility constraints,

$$J_i = \sum (z_i^* - z_i^t)^2 + \sum (y_i^* - y_i^t)^2 + \sum (y_{j_i}^* - y_{j_i}^t)^2 + \sum (x_{i_{\text{obj}}}^* - x_{i_{\text{obj}}}^t)^2. \tag{8}$$

Each discipline optimization minimizes the discrepancy between the system-level target variables and the corresponding discipline variables. The design variables consist of all variables required by the discipline, including global and local design variables, as well as non-local coupling variables. Only constraints depending on either the local design variables or local coupling variables are imposed by the discipline optimizers.

Ideally, a semi-analytic sensitivity analysis method should be used when the optimizer requires gradients. However, because the number of design variables and constraints has been reduced relative to the original problem, the use of either finite differences or the complex-step method may be acceptable.

At the system level, the choice of sensitivity analysis method significantly affects the performance of the method as a whole. Since each system-level constraint consists of multiple discipline optimizations, it is desirable to minimize the number of constraint evaluations. To this end, the post-optimal sensitivities of J^* can be computed as suggested by Braun et al. (1993),

$$\frac{\mathrm{d}J}{\mathrm{d}x_s} = \frac{\partial J}{\partial x_s} + \lambda_i \frac{\partial c_i}{\partial x_s}, \tag{9}$$

where $x_s \equiv (z, y, x_{\text{obj}})$ is the set of system-level optimization variables, and λ_i represents the Lagrange multipliers associated with discipline-level constraint, c_i. In the particular formulation of CO we used, no system-level variables have an effect on the discipline constraints. Therefore, the second term in the post-optimality equation is zero, leaving only the term that depends on the objective.

One of the major advantages of CO over monolithic approaches is that it mimics the organization of large industrial design groups, which are typically divided into different discipline design groups. The chief designer can be regarded as the system-level optimizer. The relative autonomy of each discipline allows each subproblem to be solved in parallel.

From the system-level perspective, it is largely irrelevant which optimization algorithms are used to solve the discipline subproblems, as long as the values of the compatibility constraint, the optimal design point, and sensitivities are provided.

In spite of these advantages, CO is not without its drawbacks. As the number of coupling variables increases, the dimensionality of the system-level problem increases, as does the number of variables involved in the calculation of the system-level compatibility constraints. Therefore, CO tends to be most effective in problems with few coupling variables.

Due to the form of the quadratic penalty function, CO exhibits a singular Jacobian matrix of the system-level compatibility constraints at the optimum (DeMiguel and Murray 2000, 2006). Because the value of the compatibility constraints and their gradients are both zero at solutions, the optima returned by CO are irregular optimization points. This causes the Lagrange multipliers associated with the compatibility constraints to tend to zero, resulting in numerical problems that adversely affect convergence when using gradient-based optimizers. The existence of multiple subspace solution regions can also produce inaccuracies in the system-level Jacobian (Braun et al. 1996) and further hinder convergence. Sobieski and Kroo (2000) suggested using response surfaces to model the discipline optimizations as a solution to some of the system-level convergence difficulties.

3.5 Concurrent subspace optimization

Concurrent subspace optimization (CSSO) (Sobieszczanski-Sobieski 1988; Wujek et al. 1997) is a bilevel, non-hierarchic method. In the variant used herein, a quadratic response surface is used to provide non-local information to each discipline subspace optimization, as proposed by Sellar et al. (1996). The response surface is initialized by completing an MDA at a predetermined number of design points. This allows it to provide an estimate of each discipline's state variables for any set of design variables.

The CSSO system-level optimization problem can be stated as,

$$\begin{aligned} &\text{minimize:} && f(z, x, \tilde{y}) \\ &\text{w.r.t.:} && z, x \\ &\text{s.t.:} && c(z, x, \tilde{y}) \leq 0, \end{aligned} \tag{10}$$

where $\tilde{y}$ represents the coupling variables given by the response surface approximation. The CSSO subspace optimization problem is as follows,

$$\begin{aligned} &\text{minimize:} && f(z_i, z_0, x_i, x_0, y_i(z_i, z_0, x_i, x_0, \tilde{y}_j), \tilde{y}_j) \\ &\text{w.r.t.:} && z_i, x_i \\ &\text{s.t.:} && c(z_i, z_0, x_i, x_0, y_i(z_i, z_0, x_i, x_0, \tilde{y}_j), \tilde{y}_j) \leq 0, \end{aligned} \tag{11}$$

where z_i represents the global variables assigned to discipline i, respectively; z_0 and x_0 represent the set variables held constant through the subspace optimization; and $\tilde{y}_j$ represents the non-local coupling variables obtained from the response surface approximation.

Since we use quadratic response surfaces, the number of function calls needed for the initialization is proportional to the number of design variables squared. Each function call requires a solution of the MDA, so this can quickly become too costly and limit the viability of CSSO to problems with more than 20 design variables. The use of parallel computing, adaptive response surfaces, or sensitivities can assist in decreasing this cost, but for problems with thousands of variables CSSO may still prove to be impractical unless a more efficient approximation method is used.

Another shortcoming of quadratic response surfaces is that its Hessian can be negative definite, in which case the function has no minimum. This can be prevented by using a trust region approach, which limits the movements in the design space at each iteration.

The system-level optimizer is responsible for satisfying all of the constraints in the problem, both global and local. Since each objective and constraint evaluation consists in querying the response surface for the state variables, the system optimization is solved very quickly. The impact of using inefficient sensitivity analysis methods is minimal, since the cost of retrieving response surface points is very low.

Design variables are assigned to whichever discipline subspace they exert the most influence over. For example, a global variable that affects one discipline linearly and another quadratically would be assigned to the discipline it affects quadratically.

Each subspace optimization problem uses the response surface to gather non-local state information with local discipline analyses performed to generate local state information. As with the system-level problem, each subspace optimization must also

satisfy the global and local constraints. Any design variable not included in a subspace is held constant throughout the discipline optimization process.

The efficiency of CSSO is directly dependent on the cost of producing the response surfaces. Thus it is paramount that the initial bounds for the response surface be chosen so that the surfaces need to be regenerated as few times as possible. If this can be done and the dimensionality of the problem is such that the surface can be initialized at a reasonable cost, CSSO is a competitive method.

4 Framework description

pyMDO was developed primarily to provide a platform where various MDO methods can be easily and consistently compared for a large set of problems. The primary means for accomplishing this is the elimination of problem reformulation when solving a given multidisciplinary problem in different methods. Since each MDO method produces a different set of optimization problems for a given multidisciplinary problem, researchers traditionally have had to create each of these sets of optimization problems manually.

pyMDO was programmed in Python (Langtangen 2004), which allows it to interface with analysis and optimization codes not only in Python, but more importantly in Fortran, C and C++ (Martins et al. 2008).

Object-oriented practices, such as inheritance, are used to carry forward common attributes from the optimization problem formulation to the multidisciplinary problem formulation and finally method-specific problem formulation. This makes it easier to implement, maintain and develop the various MDO methods.

Once a problem has been described in the standard form, users can select the desired MDO method using a command line option. pyMDO then casts the multidisciplinary problem into its method-specific form and creates one or more discipline optimization problems. Figure 1 shows how object-oriented programming automatically implements of each method. The optimization problem class, for example is used multiple times and in some cases it is partially changed using inheritance and overloading to adapt it to the appropriate role. For this work, the numerical optimizer used in the optimization problem class is SNOPT (Gill et al. 2002). A more complete description of pyMDO has been published in another article (Martins et al. 2008).

A simple example of how to describe a problem is shown in Fig. 2. This Python source code corresponds to the problem described in Sect. 5.1. Note that for brevity, only one of the two disciplines is defined.

5 Evaluation

To perform a comprehensive evaluation of the various MDO methods, it is desirable to test them on numerous problems with varying characteristics. In this evaluation we solve four MDO problems, the last of which is scalable.

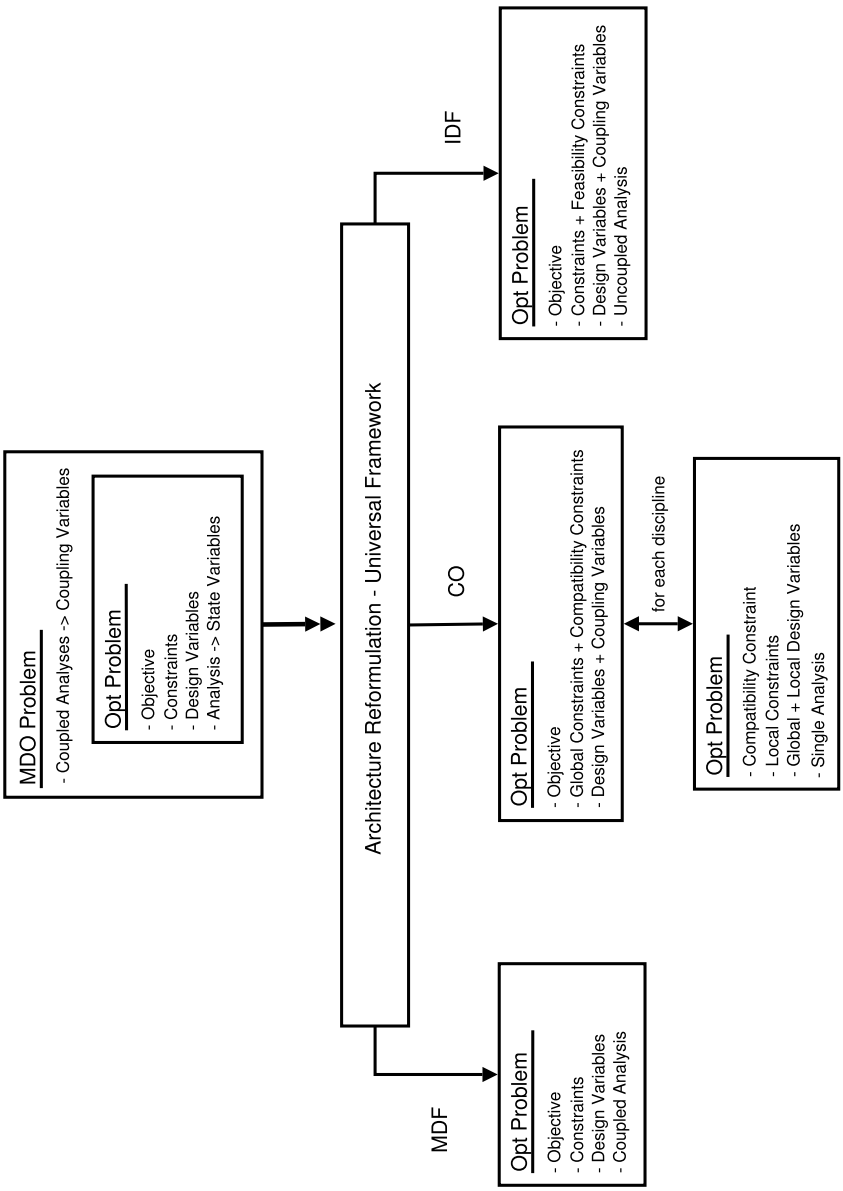

Fig. 1 Object-oriented design of pyMDO

5.1 Analytic problem

This problem has been previously solved by Sellar et al. (1996) and was selected as a first test case due to its simplicity. Although is has low dimensionality, the problem exhibits characteristics of larger MDO problems and allowed each of the method implementations to be verified prior to further testing.

The optimization problem is defined as follows:

$$\begin{aligned} &\text{minimize:} && x_1^2 + x_2 + y_1 + \mathrm{e}^{-y_2} \\ &\text{w.r.t.:} && z_1, x_1, x_2 \\ &\text{s.t.:} && 1 - y_1/3.16 \leq 0 \\ & && y_2/24 - 1 \leq 0 \\ & && -10 \leq z_1 \leq 10 \\ & && 0 \leq x_1 \leq 10 \\ & && 0 \leq x_2 \leq 10. \end{aligned} \tag{12}$$

Discipline 1 is given by,

$$y_1(z_1, x_1, x_2, y_2) = z_1^2 + x_1 + x_2 - 0.2y_2, \tag{13}$$

and Discipline 2 by,

$$y_2(z_1, x_2, y_1) = \sqrt{y_1} + z_1 + x_2. \tag{14}$$

The global optimum of this problem is located at $(z_1, x_1, x_2) = (1.9776, 0, 0)$ and has an objective value of 3.18339. The constraint in Discipline 1's constraint is active at the optimum.

Each of the two disciplines has one state variable. Each of these state variables is also a coupling variable. There are two global design variables and an additional local design variable in Discipline 1. The coupling between the two disciplines is nonlinear and each discipline has a local constraint associated with its state.

This problem was solved successfully using all five methods from a number of starting points. Under-relaxation was not required to converge the MDA. Response surface limits were initialized to the upper and lower bounds of the design variables. The convergence tolerance for the MDA module was set to 10^{-15}. All of SNOPT's parameters were left at their default values and therefore the optimality conditions were satisfied to a tolerance of 10^{-6}.

Table 1 presents the computational performance of each of the methods starting from $(z_1, x_1, x_2) = (1, 5, 2)$ using both finite differences and the complex-step method (Martins et al. 2003) to compute the necessary sensitivities. A step of 10^{-8} was used for the finite-differences and the value for the complex-step method was 10^{-20}.

Since an analytic solution can be obtained for this problem, the optimum returned by the optimizer was compared to the exact one. The comparison was made using an l^2-norm of the difference between the optimal design variables and their corresponding exact values, i.e.,

$$\text{error}_x = \|x - x_{\text{exact}}\|_2 \,. \tag{15}$$

```
# Import universal framework module
import mdoprob
# Problem setup dv_groups = {'x1':1, 'x2':1, 'x3':1}
discipline_groups = {'Discipline␣1':  {'y1':1}, \
                     'Discipline␣2':  {'y2':1}  }
objective_discipline = 'Objective'
# Architecture selection -- MDF, IDF, SAND, CO, CSSO available.
prob = mdoprob.CO(dv_groups, discipline_groups, objective_discipline)
# Set options
prob.name = "Analytic"
prob.sens_type = "CS"
prob.fpi_update = 1.0
prob.fpi_convergence_tol = 1.e-15
prob.cs_convergence_tol = 1.e-4
prob.coupling_init_type = "Automatic"
prob.RS_precision = 0.01
# Define objective function
def eval_objective(vars):
    return array([vars['x2'][0]**2 + vars['x3'][0] + vars['y1'][0] + \
                  exp(-vars['y2'][0])])
# Define objective dependencies
prob.discipline['Objective'].inputs.extend(['x2', 'x3', 'y1', 'y2'])
prob.discipline['Objective'].analysis =  eval_objective
# Discipline 1 function definitions
def dis1_analysis(vars):
    return array([vars['x1'][0]**2 + vars['x2'][0] + vars['x3'][0] - \
                  0.2*vars['y2'][0]])
def dis1_get_coupling(vars, state_vars):
    vars['y1'][:] = state_vars[:]
def dis1_constraints(vars, state_vars):
    return array([vars['y1'][0] / 3.16 - 1.0])
def dis1_residuals(vars, state_vars):
    return array([vars['x1'][0]**2 + vars['x2'][0] + vars['x3'][0] - \
                  0.2*vars['y2'][0] - state_vars[0]])
# Discipline setup - Only discipline 1 is shown
prob.discipline['Discipline␣1'].inputs.extend(['x1', 'x2', 'x3', 'y2'])
prob.discipline['Discipline␣1'].local_vars.extend(['x2'])
prob.discipline['Discipline␣1'].constraint_type = ['>']
prob.discipline['Discipline␣1'].analysis = dis1_analysis
prob.discipline['Discipline␣1'].get_coupling = dis1_get_coupling
prob.discipline['Discipline␣1'].constraints = dis1_constraints
prob.discipline['Discipline␣1'].eval_residual = dis1_residuals
# Setup design variables - Only variable x1 shown
prob.vars['x1'].value[0] = 1 prob.vars['x1'].lower[0] = -10
prob.vars['x1'].upper[0] = 10
prob.vars['x1'].influence.append('Discipline␣1')
prob.vars['x1'].RSLowerLimit[0] = -10
prob.vars['x1'].RSUpperLimit[0] = 10
# Solve the problem and retrieve results
prob.optimize()
print prob.result
print prob.obj_value
```

Fig. 2 Python source code for the analytic problem (12)

Table 1 Analytic computational performance

Architecture	Finite difference		Complex step	
	Discipline 1	Discipline 2	Discipline 1	Discipline 2
MDF	346	346	238	238
IDF	61	61	55	55
SAND	$1+57^{*}$	$1+57^{*}$	$1+49^{*}$	$1+49^{*}$
CO	1291	729	1079	587
CSSO	1250	1188	1210	1148

*Indicates residual evaluations

Table 2 l^2-norm of absolute error in optimal design variables

Architecture	Finite difference	Complex step
MDF	1.1515×10^{-6}	1.1506×10^{-6}
IDF	2.0827×10^{-9}	2.0803×10^{-9}
SAND	7.1353×10^{-7}	7.1356×10^{-7}
CO	6.1643×10^{-6}	7.3389×10^{-6}
CSSO	7.1386×10^{-6}	3.9559×10^{-6}

Table 2 shows the error for the various methods, which is within the specified tolerance for all cases.

To compare convergence histories, we used the relative error of the objective function value,

$$\varepsilon_f = \left| \frac{f - f_{\text{exact}}}{f_{\text{exact}}} \right|. \tag{16}$$

The convergence histories for the various methods are shown in Fig. 3. For MDF, IDF, SAND, and CO methods, times were recorded for each objective function call by the system-level optimizer. For CSSO, the time was recorded at the beginning of each system-level iteration and before and after each response surface generation. Flat sections in the convergence history of CSSO represent the time taken to generate (or regenerate) the response surface. A number of runs were completed with each method to ensure that the execution time profile of each method was accurate.

Note that as the overall convergence trends were identical for both the finite difference and complex-step methods, only convergence results from the finite-difference method are presented.

5.2 Speed reducer problem

This problem was adapted from the NASA MDO test suite (Alexandrov and Kodiyalam 1998; Padula et al. 1996; Kodiyalam 1998) and represents the design of a gearbox. The objective is to minimize the weight of the gearbox subject to a number of design constraints. Since the problem was originally solved as a single-discipline optimization problem, an MDO problem was created by splitting up the single discipline problem. The variable names were changed to be consistent with the conventions used in this article. The values for the constants (C) are listed in Table 3.

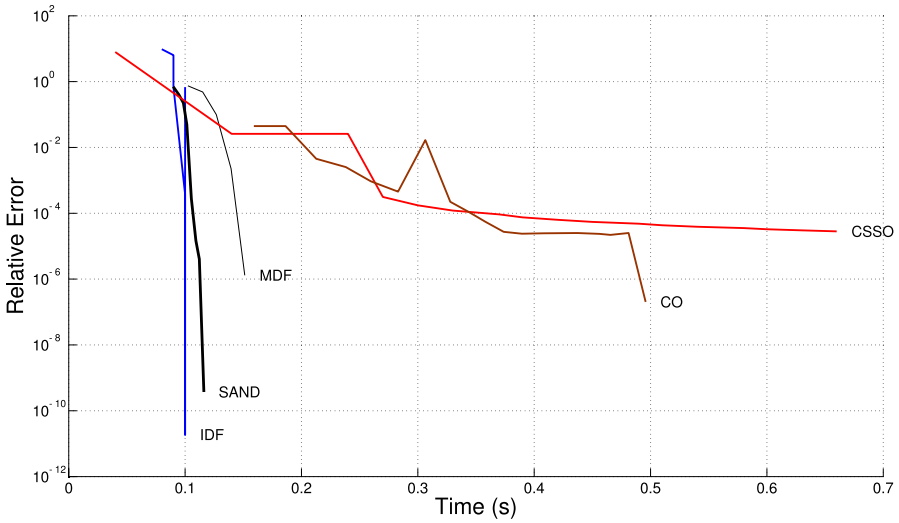

Fig. 3 Analytic problem convergence history

The speed reducer optimization problem is as follows,

$$\begin{aligned}
\text{minimize:} \quad & C_1 y_1 z_1^2 \left(C_2 z_2^2 + C_3 z_2 - C_4\right) - C_5 \left(y_2^2 + y_3^2\right) y_1 \\
& + C_6 \left(y_2^3 + y_3^3\right) + C_1 \left(x_{21} y_2^2 + x_{31} y_3^2\right) \\
\text{w.r.t.:} \quad & z_1, z_2, x_{21}, x_{31} \\
\text{s.t.:} \quad & 1 - z_1 x_3 / C_7 \geq 0 \\
& 0.7 \leq z_1 \leq 0.8 \\
& 17 \leq z_2 \leq 28 \\
& 7.3 \leq x_{21} \leq 8.3 \\
& 7.3 \leq x_{31} \leq 8.3.
\end{aligned} \tag{17}$$

Discipline 1 returns y_1, which is computed as follows,

$$\begin{aligned}
\text{find:} \quad & y_1(z_1, z_2) = \max(g_1, g_2, g_3, g_4) \\
\text{s.t.:} \quad & 1 - x1/(C_8 z_1) \geq 0 \\
& 1 - x1/C_9 \geq 0,
\end{aligned} \tag{18}$$

where,

$$g_1 = C_{10}/(z_1^2 z_2), \qquad g_2 = C_{11}/(z_1^2 z_2^2), \tag{19}$$

$$g_3 = C_{12} z_1, \qquad g_4 = C_{13}. \tag{20}$$

Table 3 Constants for the speed reducer problem

C_1	0.7854	C_{15}	1.5
C_2	3.3333	C_{16}	1.9
C_3	14.9334	C_{17}	1.93
C_4	43.0934	C_{18}	1100
C_5	1.5079	C_{19}	0.1
C_6	7.477	C_{20}	1.69×10^7
C_7	40	C_{21}	745
C_8	12	C_{22}	2.9
C_9	3.6	C_{23}	5.5
C_{10}	27	C_{24}	1.1
C_{11}	397.5	C_{25}	1.93
C_{12}	5	C_{26}	850
C_{13}	2.6	C_{27}	1.575×10^8
C_{14}	3.9	C_{28}	5

Discipline 2 solves the following problem,

$$\begin{aligned} \text{find:} \quad & y_2(z_1, z_2, x_{21}) = \max(g_5, g_6, g_7) \\ \text{s.t.:} \quad & 1 - y_2/C_{14} \geq 0 \\ & 1 - y_2 C_{15} C_{16}/x_{21} \geq 0, \end{aligned} \tag{21}$$

where,

$$g_5 = \left(C_{17} x_{21}^3 / z_1 z_2\right)^{1/4}, \tag{22}$$

$$g_6 = \left(1/C_{28} C_{19} \sqrt{C_{20}^2 x_{21}^2/(z_1^2 z_2^2) + C_{21}}\right)^{1/3}, \tag{23}$$

$$g_7 = C_{22}. \tag{24}$$

Discipline 3 is defined as,

$$\begin{aligned} \text{find:} \quad & y_3(z_1, z_2, x_{31}) = \max(g_8, g_9, g_{10}) \\ \text{s.t.:} \quad & 1 - y_3/C_{23} \geq 0 \\ & 1 - y_3 C_{24} C_{16}/x_{31} \geq 0, \end{aligned} \tag{25}$$

where,

$$g_8 = \left(C_{25} x_{31}^3 / z_1 z_2\right)^{1/4}, \tag{26}$$

$$g_9 = \left(1/(C_{26} C_{19}) \sqrt{C_{20}^2 x_{31}^2/z_1^2 z_2^2 + C_{27}}\right)^{1/3}, \tag{27}$$

$$g_{10} = C_{28}. \tag{28}$$

The optimum is at $(z_1, z_2, x_{21}, x_{31}) = (0.7, 17, 7.3, 7.7153199)$, where the objective value is 2994.355026.

Table 4 Function calls for the speed reducer problem

Architecture	Finite difference			Complex step		
	Disc. 1	Disc. 2	Disc. 3	Disc. 1	Disc. 2	Disc. 3
MDF	132	132	132	120	120	120
IDF	56	56	56	50	50	50
SAND	$1+55^*$	$1+55^*$	$1+49^*$	$1+49^*$	$1+49^*$	$1+49^*$
CO	2730	4342	3852	1730	3186	2698
CSSO	116	90	102	102	80	88

*Indicates residual evaluations

Table 5 l^2-norm of absolute error in optimal design variables

Architecture	Finite difference	Complex step
MDF	1.0377×10^{-5}	1.0378×10^{-5}
IDF	2.6167×10^{-6}	2.6172×10^{-6}
SAND	2.6170×10^{-6}	2.6164×10^{-6}
CO	3.7643×10^{-4}	9.6707×10^{-4}
CSSO	2.6036×10^{-7}	2.6004×10^{-7}

Like the analytic problem, the speed reducer problem was solved successfully using all five methods from various starting points. Under-relaxation was not required to converge the MDA and response surface limits were initialized to the upper and lower bounds of the design variables. All other optimization parameters are the same used in the analytic problem. Table 4 presents the computational performance of each of the methods starting from $(z_1, z_2, x_{21}, x_{31}) = (0.75, 22, 7.8, 8.3)$ with both finite differences and the complex-step method used to compute the sensitivities.

As with the analytic problem, the exact solution of the speed reducer problem was computed. Table 5 shows the l^2-norm of the difference between the optimal design variables and the exact values for each of the methods. The rate of convergence of the objective function for the complex-step case is shown in Fig. 4.

As can be seen from Tables 4 and 5, CO performs substantially worse than any of the other methods in solving this problem. This is related to the current CO implementation, where the system level and discipline level optimizations attempt to adjust a single value in opposite directions. This drastically slows the convergence rate of the objective as well as increasing the number of function evaluations. It also contributes to the lower resolution on the optimal variable sets returned by CO with this problem.

Two other trends are important to note with this problem. The first is that number of function evaluations saved by using the complex-step method in this case is much smaller than similar gains made with the analytic problem. Comparing the gains made with the MDF method, the speed reducer shows only a 10% reduction in the function calls made to each discipline, whereas in the analytic problem the reduction was 33%. This shows that although more accurate sensitivities can provide gains in some cases, the reduction in function evaluations that can be achieved is problem dependent.

The second important trend is the outstanding performance of CSSO in solving this problem. This can be attributed to the fact that each of the design variables has

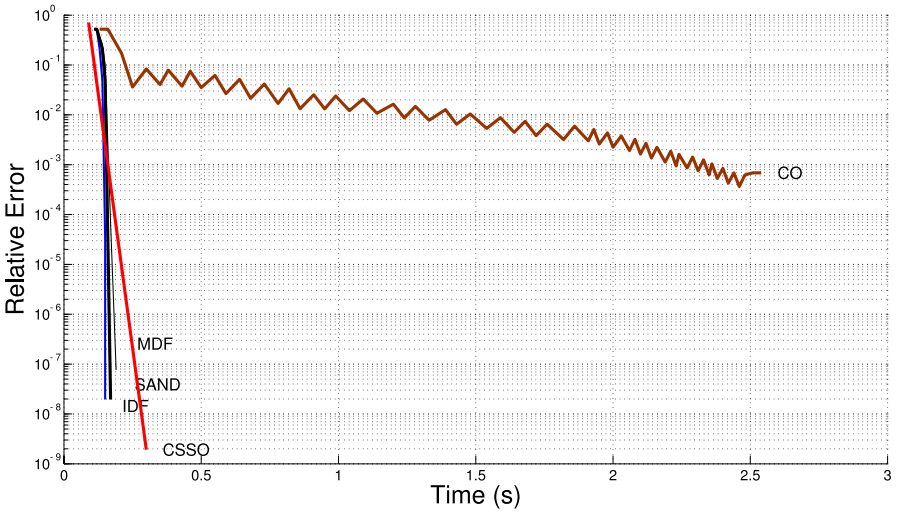

Fig. 4 Speed reducer convergence history

strictly defined limits and that the governing equations are well-behaved over the entire design space. Therefore, the initial response surface is able to model the MDA accurately throughout the entire design space and no response surface regeneration is required.

5.3 Combustion of propane problem

This problem was also taken from the NASA MDO test suite and represents the chemical equilibrium reached during the combustion of propane in air. Variables are assigned to represent each of the ten combustion products as well as the sum of the products.

In the conventional solution method, eleven nonlinear equations must be solved simultaneously with eight constants representing the pressure ($p = 40$), air to fuel ratio ($R = 10$), and six empirical constants (K_5, K_6, K_7, $K_9 = 1$, and K_8, $K_{10} = 0.1$). To obtain an MDO problem, the equations are divided into three disciplines such that coupling exists between the disciplines and the evaluation of the system level objective and constraints require the solution of the coupled system of equations.

The optimization problem is as follows,

$$\begin{aligned}
&\text{minimize:} && f_2 + f_6 + f_7 + f_9 \\
&\text{w.r.t.:} && x_1, x_3, x_6, x_7 \\
&\text{s.t.:} && f_2(x) \geq 0, \qquad f_6(x) \geq 0 \\
& && f_7(x) \geq 0, \qquad f_9(x) \geq 0 \\
&\text{with:} && x_1 \geq 0, \qquad x_3 \geq 0 \\
& && x_6 \geq 0, \qquad x_7 \geq 0,
\end{aligned} \tag{29}$$

where,

$$f_2(x) = 2x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + 2x_{10} - R, \tag{30}$$

$$f_6(x) = K_6 x_2^{1/2} x_4^{1/2} - x_1^{1/2} x_6 \left(p/x_{11}\right)^{1/2}, \tag{31}$$

$$f_7(x) = K_7 x_1^{1/2} x_2^{1/2} - x_4^{1/2} x_7 \left(p/x_{11}\right)^{1/2}, \tag{32}$$

$$f_9(x) = K_9 x_1 x_3^{1/2} - x_4 x_9 \left(p/x_{11}\right)^{1/2}. \tag{33}$$

Discipline 1 computes (x_2, x_4) by satisfying the following equations:

$$x_1 + x_4 - 3 = 0, \tag{34}$$

$$K_5 x_2 x_4 - x_1 x_5 = 0. \tag{35}$$

Discipline 2 finds (x_8, x_{10}) such that

$$K_8 x_1 - x_4 x_8 \left(p/x_{11}\right) = 0, \tag{36}$$

$$K_{10} x_1^2 - x_4^2 x_{10} \left(p/x_{11}\right) = 0. \tag{37}$$

Discipline 3 determines (x_5, x_9, x_{11}) by solving,

$$2x_2 + 2x_5 + x_6 + x_7 - 8 = 0, \tag{38}$$

$$2x_3 + x_9 - 4R = 0, \tag{39}$$

$$x_{11} - \sum_{j=1}^{10} x_j = 0. \tag{40}$$

The optimum is $(x_1, x_3, x_6, x_7) = (1.378887, 18.426810, 1.094798, 0.931214)$ and the minimum objective value is zero. All system-level inequality constraints are active at this point.

This problem was also solved successfully by each of the methods implemented within the framework. To converge the MDA, the use of under-relaxation was required. For design points in the vicinity of the solution, an under-relaxation parameter of 0.7 typically resulted for successful convergence. At design points far from the solution, the problem is harder to solve and an under-relaxation parameter of 0.4 was required. Numerical difficulties also forced the convergence tolerance of the MDA module to be loosened by one order of magnitude—from 10^{-15} to 10^{-14}. All optimization parameters for SNOPT were left at their default values.

Though a number of starting points were tested, the following results were obtained by starting from $(x_1, x_3, x_6, x_7) = (2, 20, 0, 0)$. As with the analytic problem, both the finite-difference and complex-step sensitivity analysis methods were used. Since there are no design variable bounds, the initial response surface was generated to span as much of the design space around the initial design point as feasible. Table 6 shows the computational performance for each of the methods.

To compute the reference solution to this problem, IDF was used with a convergence tolerance of 10^{-15}. IDF was chosen for this purpose since it converges not

Table 6 Combustion of propane computational performance

Architecture	Finite difference			Complex step		
	Disc. 1	Disc. 2	Disc. 3	Disc. 1	Disc. 2	Disc. 3
MDF	874	874	874	809	809	809
IDF	105	105	105	97	97	97
SAND	$1+105^*$	$1+105^*$	$1+105^*$	$1+97^*$	$1+97^*$	$1+97^*$
CO	890	538	16604	–	–	–
CSSO	6048	6060	6874	5999	6007	6667

*Indicates residual evaluations

Table 7 l^2-norm of absolute error in optimal design variables

Architecture	Finite difference	Complex step
MDF	9.9551×10^{-7}	4.1839×10^{-6}
IDF	5.0323×10^{-7}	5.0339×10^{-7}
SAND	4.5552×10^{-6}	4.5548×10^{-6}
CO	3.3897×10^{-3}	6.8244×10^{-3}
CSSO	4.7507×10^{-8}	4.7503×10^{-8}

only the design variables, but also the coupling variables to the specified tolerance. The minimum objective value returned in this reference run was $\mathcal{O}(10^{-15})$, i.e., machine zero. Table 7 shows the l^2-norm of the difference between the various results and the reference ones.

The convergence history of the objective function is shown in Fig. 5.

5.4 Scalable problem

Many multidisciplinary problems do not scale well due to the inclusion of disciplines with computationally costly solutions of the governing equations. This scalable problem was designed to allow researchers to examine the effects of increasing dimensionality while keeping manageable computational requirements. This was accomplished by conceiving a problem that is scalable, that is, given an arbitrary dimensionality of any of its components, a problem is generated automatically. For a given base class, the generated problems are mathematically similar and the following parameters can be varied:

- Number of disciplines
- Number of output coupling variables associated with each discipline
- Number of local design variables associated with each discipline
- Number of global design variables
- Strength of coupling between the disciplines.

This allows to study the effect of each of these factors on the performance of the various methods. Additionally, a number of other parameters relating to the type and

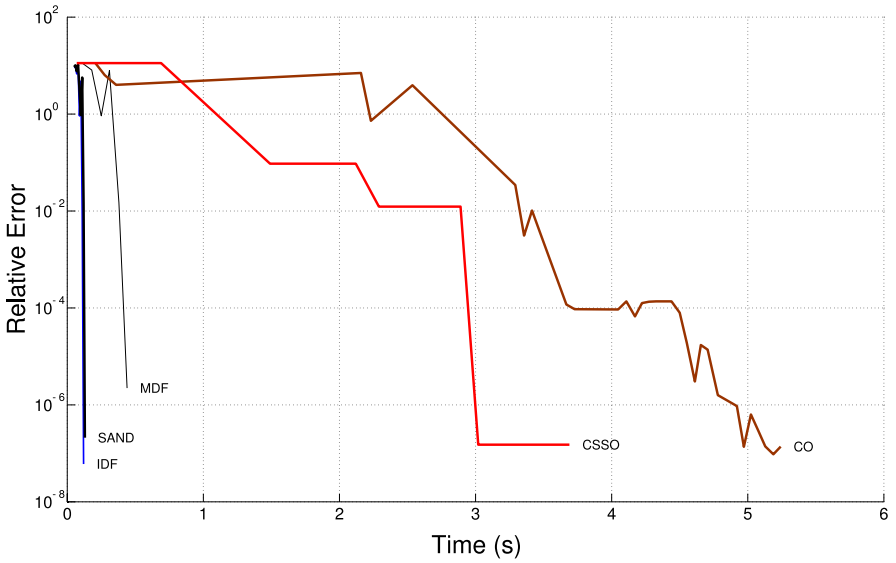

Fig. 5 Convergence history of objective for the propane problem

scope of the objective, the number and type of constraints, and the nature of the coupling can be adjusted, but are beyond the scope of this particular article.

For the scalable problem, a quadratic objective was used. The disciplines have linear dependence on each other and have local constraint on each coupling variable. The governing equations for each discipline consist of a linear system dependent on the global design variables, local design variables, and non-local coupling variables. The optimization problem statement for this problem is as follows,

$$\begin{aligned} &\text{minimize:} && z^T z + \sum_{i=1}^{N} y_i^T y_i \\ &\text{w.r.t.:} && z, x \\ &\text{s.t.:} && 1 - \frac{y_i}{C_i} \leq 0, \quad i = 1, \ldots, N. \end{aligned} \tag{41}$$

The governing equations for each discipline i are given by

$$y_i(z, x_i, y_j) = -\frac{1}{C_{y_i}}(C_z z + C_{x_i} x_i - C_{y_j} y_j). \tag{42}$$

In the above problem statement, all C's are matrices of random positive coefficients generated prior to the start of the optimization. For the purposes of this investigation all coefficients not associated with a discipline's local coupling variables have been set to unity. The local coupling variable scaling factors are set such that the scale of the problem is near unity.

Only MDF, IDF and SAND are currently available in this problem. For each scalable problem, the design variables were initialized to unity prior to the optimiza-

Table 8 Scalable problem—effect of number of design variables

Design variables per discipline:	2	20	200	2000
MDF	10348	305145	515264	–
IDF	1493	12202	30918	350782
SAND	–	–	25637	–

Table 9 Scalable problem—effect of number of coupling variables

Coupling variables per discipline:	2	20	200
MDF	55722	134653	136262
IDF	2724	9421	33934
SAND	1921	5351	–

tion. Various coupling variable initialization methods were used, depending on the method. Typically these involve the completion of an MDA. To aid the convergence of the Gauss–Seidel iterative solver, an under-relaxation parameter of 0.7 was used. The convergence tolerance of the method was 10^{-14}. Most of the SNOPT parameters were left at the default values, including the convergence tolerance on the objective and constraints, which is 10^{-6}.

Two investigations were performed for this problem. In the first one, the number of design variables was varied while keeping all other dimensions of the problem constant. In the second investigation, the number of local design variables for each discipline was varied. Each problem consisted of three disciplines and three global design variables. When fixed for the first investigation, the number of local design variables and coupling variables were set at 50 per discipline, for a total of 150. The number of function evaluations required per discipline for the two investigations is shown in Tables 8 and 9. Entries marked with a dash denote failure to converge to an optimum. For higher-dimensional problems this was typically due to a failure in the optimization algorithm.

In addition to the number of function evaluations, the total optimization time of each method was recorded and is presented in Figs. 6 and 7.

6 Concluding remarks

pyMDO was shown to be an invaluable tool for the rapid implementation of multiple MDO methods for a given problem. This makes it an ideal platform for benchmarking current and future methods.

The addition of a scalable problem to the set of pyMDO examples has allowed a great deal of information to be gained on the relative performance of a number of the MDO methods.

For the fully coupled, highly-constrained problems investigated herein, a number of conclusions can be drawn with respect to robustness and computational efficiency.

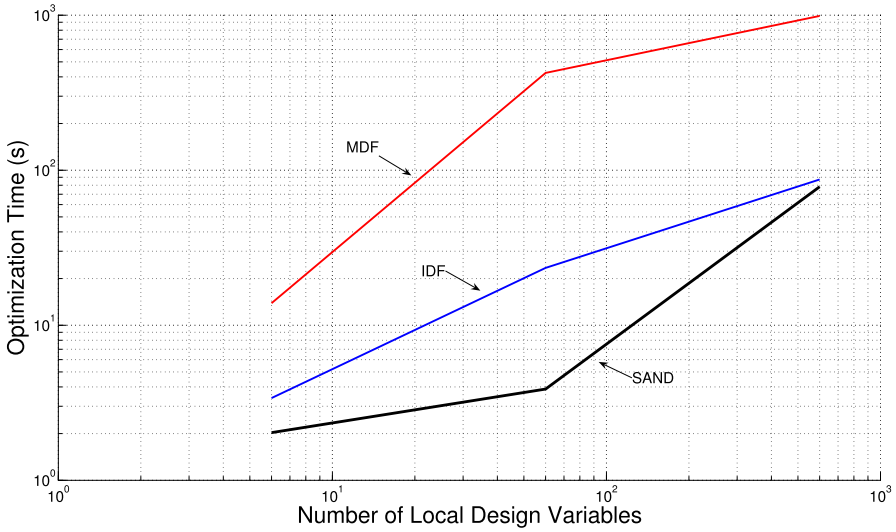

Fig. 6 Scalable problem—effect of number of design variables

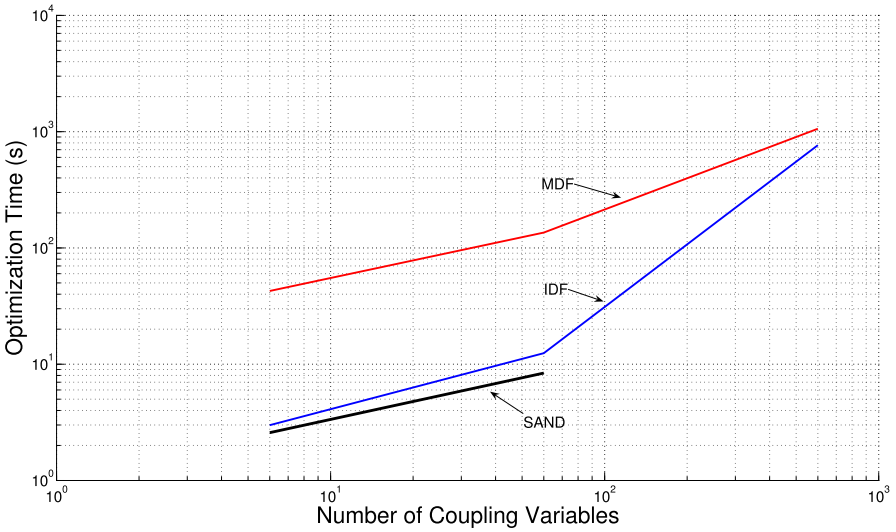

Fig. 7 Scalable problem—effect of number of coupling variables

6.1 Robustness

MDF and IDF found optima with the least number of failures, and are thus the most robust approaches for this set of problems. CSSO and other MDO methods that rely on approximation methods are largely dependent on the characteristics of the problem and the approximation method used. For the quadratic response surface used in this study, modeling in excess of 15 variables could not be done reliably. Additionally,

difficulties are routinely encountered in determining the bounds on the approximation models when no prior knowledge of the problem exists.

6.2 Computational efficiency

In both the analytic problem and the scalable problems, SAND was able to consistently outperform the other methods in terms of computational cost. Although the coupling and state variables were identical for each of the problems investigated, SAND's use of state variables and residual constraints made it converge more rapidly than IDF. In cases for which SAND was not able to find a solution, IDF was typically the most computationally efficient method. It is assumed that additional problems would reveal cases where both IDF and SAND are outperformed by either MDF or one of the bilevel methods under implementation. Upon examining Fig. 7, we can see that the slope of the MDF plot is smaller than that of IDF. Providing results can be obtained beyond 600 coupling variables, MDF may well begin to outperform IDF.

It should be noted that these results were obtained using either finite differences or the complex-step method to calculate required sensitivities. If the comparison were performed using either an adjoint or direct sensitivity analysis method for each method, it is likely that a performance comparison would result in a significantly different ranking.

Acknowledgements The authors are very grateful for the support provided by the Canada Research Chairs program and the Natural Sciences and Engineering Research Council. They are also thankful for Andrew Lambe's assistance.

References

Alexandrov NM, Kodiyalam S (1998) Initial results of an MDO evaluation survey. AIAA Paper 98-4884

Alexandrov NM, Lewis RM (1999) Comparative properties of collaborative optimization and other approaches to MDO. In: Proceedings of the first ASMO UK/ISSMO conference on engineering design optimization

Alexandrov NM, Lewis RM (2002) Analytical and computational aspects of collaborative optimization for multidisciplinary design. AIAA J 40(2):301–309

Braun RD, Kroo IM (1997) Development and application of the collaborative optimization architecture in a multidisciplinary design environment. In: Alexandrov N, Hussaini MY (eds) Multidisciplinary design optimization: state of the art. SIAM, Philadelphia, pp 98–116

Braun RD, Kroo IM, Gage PJ (1993) Post-optimality analysis in aerospace vehicle design. In: Proceedings of the AIAA aircraft design, systems and operations meeting, Monterey, CA, AIAA 93-3932

Braun RD, Gage PJ, Kroo IM, Sobieski IP (1996) Implementation and performance issues in collaborative optimization. AIAA Paper 96-4017

Brown NF, Olds JR (2006) Evaluation of multidisciplinary optimization techniques applied to a reusable launch vehicle. J Spacecr Rockets 43(6):1289–1300

Cramer EJ, Dennis JE, Frank PD, Lewis RM, Shubin GR (1994) Problem formulation for multidisciplinary optimization. SIAM J Optim 4(4):754–776

DeMiguel A-V, Murray W (2000) An analysis of collaborative optimization methods. In: Proceedings of the 8th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization, Long Beach, CA, AIAA 2000-4720

DeMiguel V, Murray W (2006) A local convergence analysis of bilevel decomposition algorithms. Optim Eng 7(2):99–133

Gill PE, Murray W, Saunders MA (2002) SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM J Optim 12(4):979–1006

Kodiyalam S (1998) Evaluation of methods for multidisciplinary design optimization (MDO), Part 1. NASA Report CR-2000-210313

Langtangen HP (2004) Python scripting for computational science. Springer, Berlin

Martins JRRA, Sturdza P, Alonso JJ (2003) The complex-step derivative approximation. ACM Trans Math Softw 29(3):245–262

Martins JRRA, Alonso JJ, Reuther JJ (2005) A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. Optim Eng 6(1):33–62

Martins JRRA, Marriage C, Tedford NP (2008) pyMDO: an object-oriented framework for multidisciplinary design optimization. ACM Trans Math Softw 36(4):1–23

Padula SL, Alexandrov N, Green LL (1996) MDO test suite at NASA Langley research center. In: Proceedings of the 6th AIAA/NASA/ISSMO symposium on multidisciplinary analysis and optimization, Bellevue, WA, AIAA 1996-4028

Perez RE, Liu HHT, Behdinan K (2004) Evaluation of multidisciplinary optimization approaches for aircraft conceptual design. In: Proceedings of the 10th AIAA/ISSMO multidisciplinary analysis and optimization conference, Albany, NY, AIAA 2004-4537

Sellar RS, Batill SM, Renaud JE (1996) Response surface based, concurrent subspace optimization for multidisciplinary system design. In: Proceedings of the 34th AIAA aerospace sciences meeting and exhibit, Reno, NV, AIAA 1996-0714

Sobieski IP, Kroo IM (2000) Collaborative optimization using response surface estimation. AIAA J 38(10):1931–1938

Sobieszczanski-Sobieski J (1988) Optimization by decomposition: a step from hierarchic to non-hierarchic systems. NASA Technical Report CP-3031

Sobieszczanski-Sobieski J, Altus TD, Phillips M, Sandusky R (2003) Bilevel integrated system synthesis for concurrent and distributed processing. AIAA J 41(10):1996–2003

Wujek B, Renaud J, Batill S (1997) A concurrent engineering approach for multidisciplinary design in a distributed computing environment. In: Alexandrov N, Hussaini MY (eds) Multidisciplinary design optimization: state of the art. SIAM, Philadelphia, pp 189–208