12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference
10 - 12 September 2008, Victoria, British Columbia Canada

AIAA 2008-5956

# Reconfigurable Semi-Analytic Sensitivity Methods and MDO Architectures within the $\pi$MDO Framework

Christopher J. Marriage[*] and Joaquim R.R.A. Martins[†]
*University of Toronto Institute for Aerospace Studies, 4925 Dufferin St., Toronto, ON, Canada, M3H 5T6*

**$\pi$MDO is a software framework which allows the automatic implementation of multidisciplinary design optimization (MDO) architectures to find the optimum of any appropriately defined multidisciplinary problem. A significant improvement and extension of the $\pi$MDO framework is presented. Using the inherent advantages of $\pi$MDO's object oriented and highly flexible structure, several semi-analytic sensitivity methods are implemented for the MDO architectures within the framework. Their generalized nature allows the application of these powerful and efficient methods to any problem defined within $\pi$MDO, without modification to the existing structure of the problem. Further, exploiting information gathered by these methods, a new "meta" MDO architecture is proposed which dynamically reconfigures the problem to speed the optimization while maintaining the fidelity of the original analysis. This hybrid approach uses the existing architectures in $\pi$MDO encapsulated within each other to reduce the dimensionality of coupling between disciplines. Again, due to the object oriented nature of $\pi$MDO, no modifications are required to the problem statement or the MDO architectures within the framework. Initial results suggest that both these additions produce valuable performance gains while maintaining the general flexibility and simplicity characteristic of $\pi$MDO.**

## Nomenclature

| | |
|---|---|
| $f$ | Function of interest (Objective function to be minimized) |
| $z$ | Global design variable (Design variable which affects one or more disciplines) |
| $x_i$ | Local design variable (Design variable which only affects a single discipline) |
| $y_j$ | Coupling variable (Outputs of a discipline analysis input to at least one other discipline) |
| $\mathcal{R}$ | Residual of a discipline governing equation |
| $c$ | Constraint (Global or local constraint to be satisfied) |
| $z^*, x^*, y^*$ | System level target variables, used in collaborative optimization |

## I. Introduction

Multidisciplinary design optimization is a growing area of research which seeks faster, and more optimal, solutions to multi-disciplinary design problems. $\pi$MDO[14] is a software framework which uses object oriented programming techniques and the mathematical reconfigurability of MDO problems to quickly apply various MDO architectures to any arbitrary multi-disciplinary problem. Presently, $\pi$MDO includes support for the user to provide sensitivities directly to the optimizer, or it can solve for these sensitivities through real or complex step finite differencing.[16] While the user could provide analytic or semi-analytic sensitivities directly to the optimizer, they would be problem and architecture dependent and detrimentally affect the simplicity and adaptability of the framework. Fortunately, the object oriented nature of $\pi$MDO allows the application of more efficient semi-analytic methods while maintaining the simple and generalised form of the framework. An additional goal of $\pi$MDO is to spur the development of improved MDO architectures and methods. Information derived from the semi-analytic methods can aid in the development of hybrid and more efficient MDO architectures that combine the benefits of the existing architectures within $\pi$MDO.

[*]M.A.Sc Candidate, AIAA Student Member
[†]Assistant Professor, AIAA Member

## I.A. Background

The semi-analytic sensitivity methods applied to $\pi$MDO reduce the computational expense of finding the sensitivities required for gradient based optimization while retaining the general applicability of finite differencing or automatic differentiation. In general, they exploit the structure of a multi-disciplinary problem to solve for the global or system level derivatives using only the partial derivatives from individual disciplines (the sub-system level). The two major forms of semi-analytic sensitivity methods applied were the Global Sensitivity Equations 1 and 2 (GSE1 and GSE2)[18] or direct methods, and the coupled adjoint method[12].[11]

### *I.A.1. Semi-Analytic Sensitivity Methods*

In general, a multidisciplinary problem can be described as the minimization of a function $f$ with respect to certain design variables as shown below. The objective function $f$ is directly dependent on the global and local design variables $z$ and $x$, and implicitly dependent on each discipline analysis $y(z,x)$. Finally, $c$ represents a constraint at either the discipline or global level, and $\mathcal{R}$ represents the residuals of the governing equations for each discipline which should be driven to zero at a multidisciplinary feasible solution.

$$\begin{array}{rc} \text{minimize} & f\left(z,x,y\left(z,x\right)\right) \\ \text{w.r.t.} & z,x \\ \text{s.t.} & c\left(z,x,y\left(z,x\right)\right)\leq 0 \\ & \mathcal{R}\left(x,y\left(z,x\right)\right)=0 \end{array}$$

The sensitivity of the objective function $f$ can be written using the chain rule as in equation 1 below. For the sake of brevity $x_n$ represents the set of design variables $z,x$ at the system level.

$$\frac{df}{dx_n}=\frac{\partial f}{\partial x_n}+\frac{\partial f}{\partial y_i}\frac{dy_i}{dx_n} \tag{1}$$

Martins[12] showed that this was analogous to equation 2 assuming a converged multidisciplinary solution (residuals are equal to zero). The semi-analytic methods applied to $\pi$MDO solve equation 2 and essentially differ only in the order in which they perform the required matrix multiplication.

$$\frac{df}{dx_n}=\frac{\partial f}{\partial x_n}+\frac{\partial f}{\partial y_i}\left[\frac{\partial\mathcal{R}}{\partial y}\right]^{-1}\frac{\partial\mathcal{R}}{\partial x_n} \tag{2}$$

For the GSE2 and non residual version of the adjoint method, the partials $\frac{\partial\mathcal{R}}{\partial y}$ are replaced with the partials $\frac{\partial y_i}{\partial y_j}$ of the discipline output coupling variables with respect to the outputs of the other disciplines.

### *I.A.2. Global Sensitivity Equations*

Sobieski[18] examined the use of discipline level partial derivatives to directly calculate system total derivatives. He showed mathematically that the partials $\frac{\partial\mathcal{R}}{\partial y}$ in equation (2) were analogous to the partial derivative of the outputs of a discipline with respect to the state variables of the other disciplines in the system $\frac{\partial y_i}{\partial y_j}$. He proposed two versions of the global sensitivity equations, GSE1 and GSE2, where GSE1 uses the partial derivatives of the discipline residuals, and GSE2 uses the partial derivatives of discipline outputs.

The implementation of GSE1 and GSE2 is identical except for the partial derivatives used, so the GSE1, or direct method[10] is explained. GSE2 can be implemented by replacing the residual term $\mathcal{R}$ with the discipline output $y_i$.

### I.A.3. *Direct Method (GSE1)*

In the direct method, the total derivative $\frac{dy_i}{dx_n}$ is calculated by solving equation 3 and then multiplied by $\frac{\partial f}{\partial y_i}$ to find the system totals as in equation 1. The direct method requires the solution of $\frac{dy_i}{dx_n}$ once for every design variable, so it is more useful when the number of output functions $k$ is greater than the number of design variables $n$.

$$\frac{\partial \mathcal{R}_i}{\partial \mathcal{R}_j}\left[\frac{dy_i}{dx_n}\right] = \frac{\partial \mathcal{R}_j}{\partial x_n} \ , \ j = 0 \mathinner{..} k \ , \ j \neq i \tag{3}$$

Within $\pi$MDO, any discipline governing equation which has an affect on another discipline is defined as a coupling variable, hence for GSE2 the partial derivatives of a discipline's output variables replace the residual partials of GSE1. In situations with iterative or computationally expensive discipline analyses, GSE1 may have an advantage over GSE2. GSE1, however, requires the existence of separate residual analysis code which must be provided by the user. GSE2 is therefore more widely applicable, whereas GSE1 may be faster in certain situations.

### I.A.4. *Coupled Adjoint Method*

In the adjoint method, the adjoint vector $\psi$ or $\frac{\partial f}{\partial y_i}\left[\frac{\partial \mathcal{R}}{\partial y}\right]^{-1}$ is first calculated by solving equation (4). $\psi$ is then substituted into equation (2) to find the total derivative $\frac{df}{dx_n}$.

$$\frac{\partial \mathcal{R}}{\partial y}\psi = -\frac{\partial f}{\partial y_i} \tag{4}$$

This requires the vector $\psi$ to be calculated only once for each function of interest, it is therefore useful in situations where the number of design variables exceeds the number of system level functions. Further, as with the GSE2 method, Martins[12] contends that this formulation would also be valid using the partial derivatives of discipline outputs $\frac{\partial y_i}{\partial y_j}$ in place of the residual based formulation of equation 2.

Both the adjoint and direct methods have the potential to reduce the computational expense of finding the total derivatives of a multidisciplinary problem. Because only partial derivatives are required by these methods, they avoid the completion of a system level analysis for every design variable or function. Further, the practitioner has considerable flexibility in calculating the partial derivatives required by these methods. Finite differencing, complex step,[13] automatic differentiation[15],[9] or even hand derived analytic methods can be used. $\pi$MDO allows users to define the partial derivatives required or can automatically calculate them using finite differencing or complex step methods. The availability of several variants of semi-analytic method maximizes their flexibility and the potential gains. For example, the direct method can be used to calculate the numerous system level constraints of a problem while the adjoint method is used for the single objective function.

### I.A.5. *MDO Architectures*

Multidisciplinary Design Optimization (MDO) architectures are a means of reorganizing a coupled multidisciplinary problem into a structure compatible with the mature single discipline optimization algorithms. In effect, the different architectures translate a coupled multidisciplinary problem to resemble one or more classical optimization problems. Two broad classes of architectures exist, monolithic and hierarchical (decompositional) architectures. Alexandrov et. al. showed that through the reconfigurability of MDO problems, all other MDO architectures can be derived from the basic problem statement[34] given above. $\pi$MDO currently includes three single level architectures: Multidisciplinary Design Feasible (MDF), Individual Discipline Feasible (IDF), and Simultaneous Analysis and Design (SAND). The $\pi$MDO framework also includes several versions of the bi-level architectures Collaborative Optimization (CO), and Concurrent Subspace Optimization (CSSO)[19].[14] Initial attempts at problem reconfiguration were focused on the MDF and CO architectures due to their unique attributes.

Multidisciplinary Design Feasible,[7] also known as All At Once (AAO),[1] is the simplest MDO architecture. It consists of solving a multidisciplinary analysis at each design point and is the most intuitive to engineers. The algorithm adjusts the design variables, selects a design point, then performs a fixed point iteration to converge the coupling variables to a multidisciplinary feasible solution. The MDF architecture minimizes the objective function of the original problem without modifications. MDF is robust and simple to implement, but is computationally expensive as a multidisciplinary analysis must be iteratively converged at each design point. If finite difference sensitivities are used, this multidisciplinary analysis must be converged once for every design variable, causing performance to deteriorate as the number of variables increases. A diagram of the MDF architecture is given in figure 1.

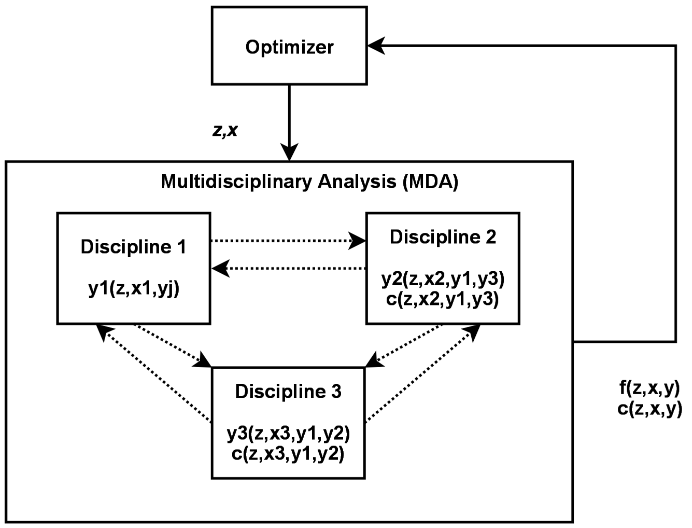

**Figure 1. Diagram of MDF architecture. The coupled system is iterated about a fixed point until multidisciplinary feasability is reached.**

Collaborative Optimization (CO) is a bi-level, decentralized architecture with optimizations occurring at both the discipline and system level.[5] The problem is decomposed into a system level optimization and a series of independent optimizations at the discipline level. Local feasibility is maintained at the discipline level by the sub-system optimizations as they are responsible for satisfying the local discipline constraints. Inter-disciplinary feasibility is maintained by the use of compatibility constraints. These constraints minimize the discrepancy between the system level targets and the actual values of the coupling and design variables at the optimum. Global constraints are handled by the system level optimizer. A diagram of the CO architecture is given in figure

Mathematically, the CO architecture can be described at the system level as in equation 5. The system level optimizer sets system targets for the global design variables $z^*$, local design variables which directly affect the optimum $x^*{}_{obj}$, and for the coupling variables for each discipline $y^*$. The system level problem calculates the objective value using these targets, and is responsible for satisfying the global constraints $c_{global}$, and one compatibility constraint for each discipline $J_i$.

$$\begin{aligned} \text{minimize} \quad & f\left(z^*, x^*{}_{obj}, y^*\right) \\ \text{with respect to} \quad & z^*, x^*{}_{obj}, y^* \\ \text{subject to} \quad & c_{global}\left(z^*, x^*, y^*\left(z, x\right)\right) \leq 0 \\ & J_i\left(z_i, {z_i}^*, x_i, {x_i}^*, y_j, {y_j}^*\right) = 0 \end{aligned} \tag{5}$$

The subsystem problem is defined as in equation (6). At the subsystem level, the objective is to minimize that particular discipline's compatibility constraint, as given in equation (7), while satisfying the local constraints. The subsystem optimization has authority over the local and global design variables $x_i$ and $z_i$, and also over the coupling inputs from other disciplines $y_j$. This formulation corresponds to the $CO_2$ formulation

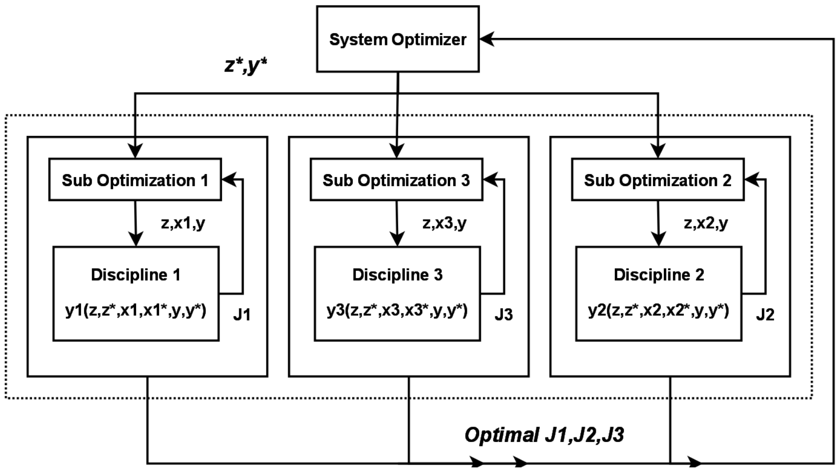

**Figure 2. CO architecture. An independant optimization problem is created for each discipline, with the system level optimization setting targets for the coupling variables of the sub-optimizations.**

presented in[5] which uses a quadratic penalty function as the compatibility constraint.

$$\begin{aligned} \text{minimize} \quad & J_i\left(z_i, {z_i}^*, x_i, {x_i}^*, y_j, {y_j}^*\right) \\ \text{with respect to} \quad & x_i, z_i, y_j \\ \text{subject to} \quad & c\left(x_i, z_i, y_i\left(x_i, y_j, z_i\right)\right) \leq 0 \end{aligned} \tag{6}$$

Where the compatibility constraint for each discipline $J_i$ is calculated as:

$$J_i = \sum (z_i - z_i^*)^2 + \sum (x_i - {x_i}^*)^2 + \sum (y_i - y_i^*)^2 + \sum (y_{j_i} - y_{j_i}^*)^2 \tag{7}$$

As the individual optimizations in CO are decoupled from the system level problem, the CO architecture performs well on distributed problems with low coupling between disciplines. Unfortunately, the output coupling variables from each discipline are part of both the system level and sub-system level objectives, so as the dimensionality of the coupling between disciplines increases, the perfomance of the architecture deteriorates. CO works best for MDO problems with relatively complex disciplinary analyses and low numbers of coupling variables between disciplines.

### *I.A.6. Reconfigurability*

A unique advantage of the object oriented form of the $\pi$MDO framework is that individual components can be reorganized and instantiated at will. Further, this structure extends to the MDO architectures it implements, allowing multiple architectures to be instantiated simultaneously and for hybrid architectures to be quickly developed and implemented. In line with the overriding objective of $\pi$MDO, this will require no changes to problem statement or user inputs, save for the ability to specify which disciplines or architectures to group together.

## II. Methodology

The application of these semi-analytic methods to $\pi$MDO uses object oriented programming and inheritance to maximize the flexibility of the resulting software. Methods to calculate the partial derivatives, combine them into the semi-analytic methods, and finally apply them to specific MDO architectures were incorporated into the existing structure of $\pi$MDO. Further, as even the architectures themselves are objects within the framework, their reconfigurability was explored by combining existing architectures and encapsulating them within each other.

### II.A. Semi-Analytic Methods

The semi-analytic methods were applied according to the theory described in section I. In line with the philosophy of $\pi$MDO, the major tasks required by these methods were incorporated as functions within the existing class structure and made use of inheritance.[20] As shown in figure 3, the routine which calculates the partial derivatives is contained within the MdoProb class. This routine has provisions to accept user provided partial derivatives or use the finite difference or complex step method[16] to calculate them. Due to its greater accuracy, the complex step approximation is preferred if the problem supports complex analysis. Because the finite differencing only incorporates the partials, rather than the total derivatives, it takes significantly less time. More significantly, this process can be run in parallel on many processors, dramatically reducing the retrieval time for the partial derivatives. Also contained in the MdoProb class are the basic routines which assemble the partial derivatives and calculate a solution of either the adjoint or direct methods. These routines will be inherited by every MDO architecture within $\pi$MDO, and those that exploit the semi-analytic methods will call specific methods from the MdoProb level. Depending on the architecture, different combinations of adjoint and direct methods will be used to calculate the total derivatives to be passed to the optimizer. In the final step, the routine at the architecture level will overwrite the existing sensitivities provided to the optimizer within the OptProb class - in this way the original structure, and adaptability, of the framework is largely retained while the efficiency of the sensitivity routines is increased.

To maximize the flexibility of the sensitivity methods, the routines which gather the partial derivatives were designed to allow user provided derivatives for any discipline analysis or constraint. More importantly, the routines detect which sensitivities have been provided by the user, and calculate any missing derivatives with finite differencing or the complex step. This allows users to provide all of the derivatives, some, or none, with the $\pi$MDO framework automatically gathering any missing information. Further, the partial derivatives are input as a python dictionary keyed by variable name. As long as the derivatives are properly labelled in the dictionary, no set input order or format is required from the user. This allows the practitioner to provide as many derivatives as they have, in whatever order they want, without sacrificing the generality and robustness of the semi-analytic methods overall.

### II.B. New and modified architectures

To create new higher level architectures, the tightly coupled disciplines of the original problem were combined into a new "super" discipline object at the system level. When the analysis of this new discipline is invoked, instead of simply performing an analysis, a multidisciplinary optimization will take place using one of the existing architectures within $\pi$MDO. Crucially, at the system level the optimizer will only see a black box, not the nested optimization occurring beneath it, so the structure of the existing architectures in $\pi$MDO can remain unchanged even as they are combined and reconfigured. Using this new architecture as a "meta" architecture to arrange and implement other architectures, the benefits of reconfigurability can be explored with different combinations of architectures, nested optimizations, and reduced coupling at the system level. Initial efforts focused on a combination of MDF, for tightly coupled disciplines, and collaborative optimization[28] for the system level, but many combinations are possible and can be explored in future.

#### *II.B.1. NEST Architecture*

In aerospace applications of MDO, difficulty is encountered in solving large problems with many disciplines and variable degrees of coupling between these disciplines. Bi-level architectures like Collaborative Optimization perform well for problems with low-fidelity coupling between disciplines and relatively complex discipline level routines. If several disciplines are highly coupled, however, the performance of CO is reduced at a geometric rate. For tightly coupled problems with few disciplines, monolithic architectures such as MDF, especially with semi-analytic sensitivites, perform well. The proposed nested optimization (NEST) architecture combines the attributes of these two architectures to produce a formulation that works for widely distributed problems with clusters of high local coupling dimensionality. In effect, it extends and

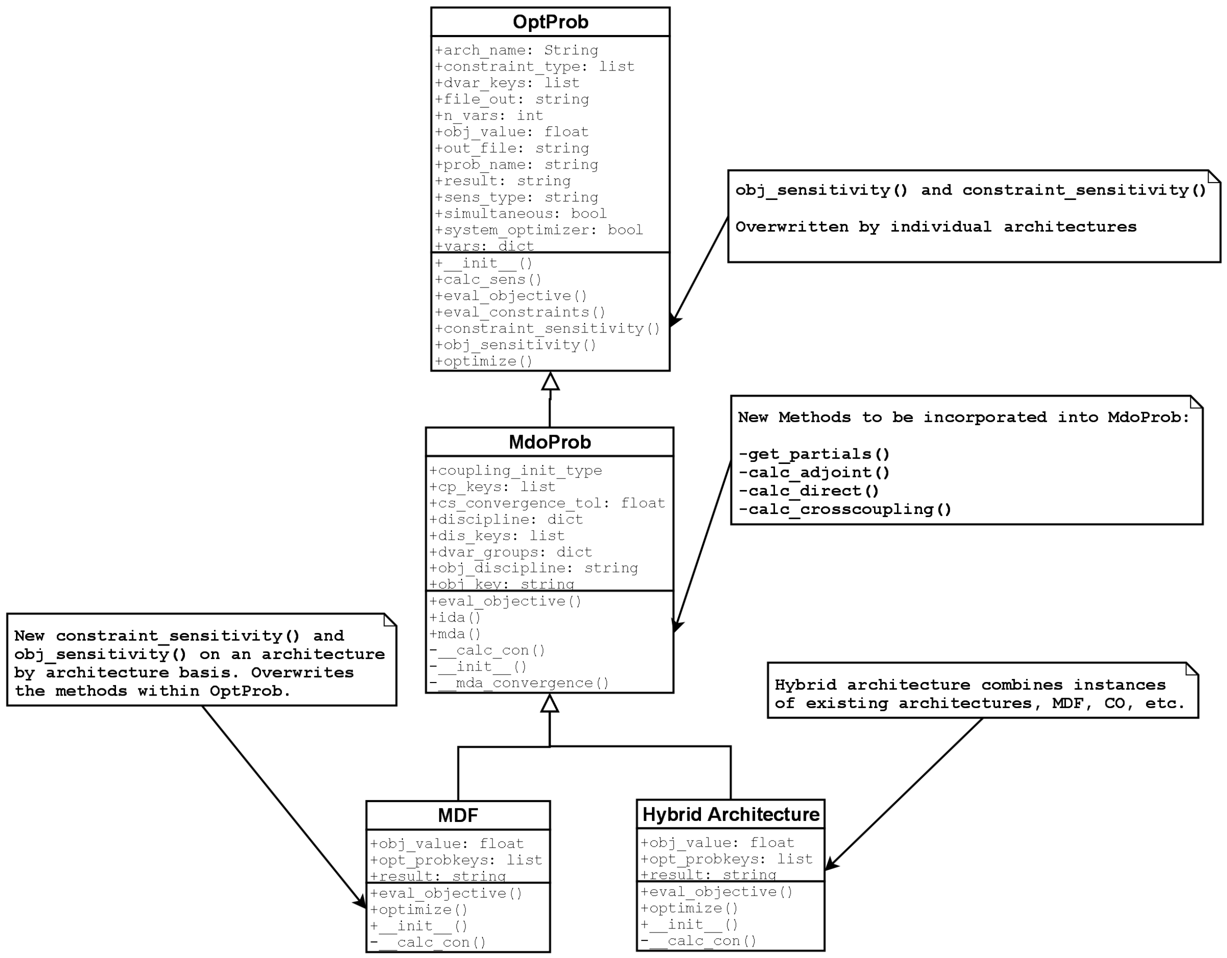

**Figure 3. Concise UML diagram showing the modifications and additions planned for $\pi$MDO. Most significantly, the new semi-analytic sensitivity methods are incorporated into the existing classes to maximize their flexibility.**

automates the original philosophy of collaborative optimization, allowing the practitioner to optimize each discipline in the most efficient way.

Structurally, the NEST architecture exploits the object oriented nature of $\pi$MDO. As the variables, disciplines, and even architectures within the framework all function as individual entities, they can be reordered and recombined with a minimum of effort. The essential tactic of the NEST architecture is to modify the discipline structure of the original problem by creating new "super" disciplines that contain instances of the MDF architecture. Note that the problem statement input to $\pi$MDO requires no changes except for a dictionary indicating which disciplines should be clustered together. In future, this choice can be automated based on ratios of coupling to design variables and interpretation of the global sensitivity equations. The reconfiguration for a sample problem is illustrated in figure 4.

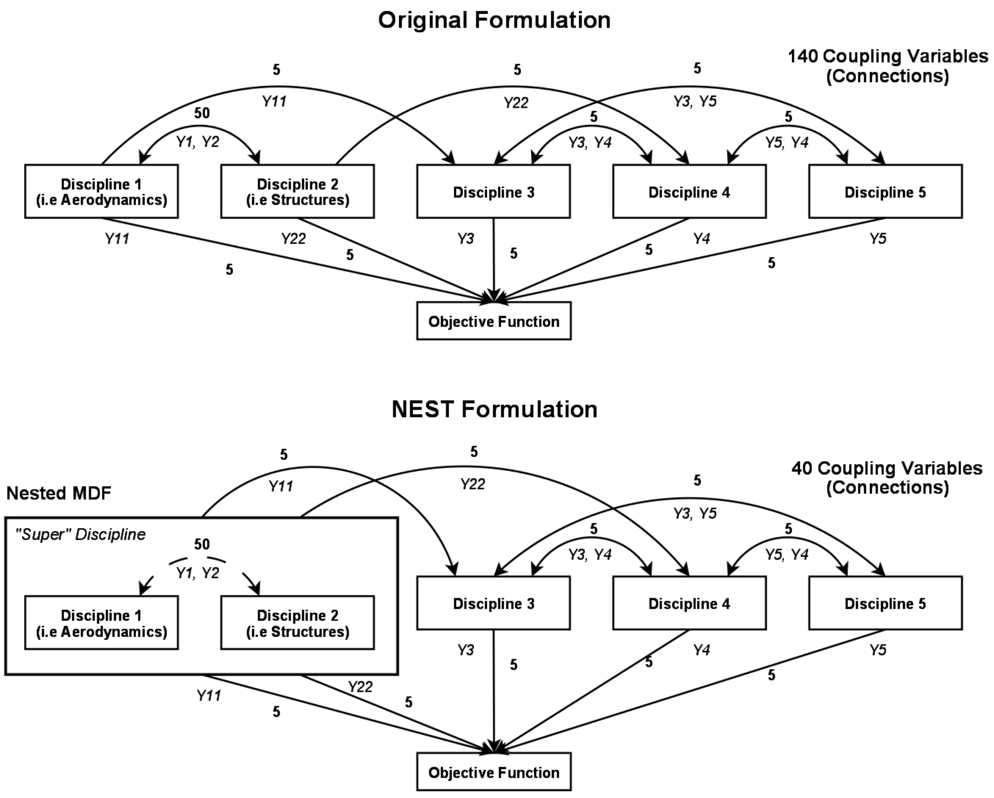

**Figure 4. Problem Reconfiguration of the NEST architecture. Tightly coupled disciplines are clustered together in "super" disciplines to reduce system level coupling dimensionality.**

The new super disciplines encapsulate an MDF architecture with a slightly modified objective function. This sub-optimization uses the multidisciplinary analysis and constraint calculations from the MDF formulation with a modified objective function taken from CO. The sub-optimization therefore attempts to minimize the discrepancy between its output coupling variables and the sytem level targets while satisfying multidisciplinary feasibility and discipline level constraints for the sub disciplines. More robust convergence is obtained by using a slightly modified CO formulation, proposed by Perez,[17] which makes the compatibility constraints inequality constraints at the system level. By slightly relaxing the tolerances of these inequality constraints at the system level, the CO architecture converges faster and more robustly while maintaining comparable accuracy.

Compared to a normal CO formulation, many coupling variables are removed from the system level problem, speeding convergence. Compared to a monolithic MDF formulation, there is now autonomy between most discipline analyses, and the fixed point iteration encompasses fewer disciplines and variables. Using the CO sub-system objective function as the objective of the MDF sub-optimizations allows the unmodified post-optimality sensitivity methods from the CO architecture to be used. This speeds implementation and avoids problem dependant calculation of second order derivatives, required when multilevel optimization problems are formulated.[6]

## III. Results

To test the performance of these new methods, a variety of experiments were performed. First, a study was conducted of the relative computational cost of the adjoint and direct methods and parameters were developed to guide their application. The performance of the semi-analytic methods applied to the MDF architecture was then tested for different problem attributes. This was accomplished using a scalable test problem which allows an arbitrary number of disciplines and variable dimensionality of local and coupling variables.[20] Finally, the reconfigurable architecture was tested on a version of the scalable problem tailored to have appropriate coupling attributes.

### III.A. MDF Implementation

For the initial implementation, the MDF architecture was modified to make use of a coupled adjoint formulation to calculate the sensitivities of the objective function, and a GSE1 direct method for the sensitivities of the constraint functions. In the final implementation, the choice of which semi-analytic method to apply will not be arbitrary and will depend on the relative dimensionality of the constraint functions versus the number of design variables. Even in the initial implementation, however, this approach showed a great improvement over the original formulation. Depending on the specific number of local and coupling variables, the advantage varied, but was always significant. A representative plot of the sensitivity calculation times for the scalable problem is shown in figure 6 in the appendix. A study of the performance of the modified architecture with increasing dimensionality was also performed and this is shown in figure **??** in the appendix.

#### *III.A.1. Sensitivity Validation*

To validate the sensitivities calculated by the semi-analytic methods, they were compared to the results achieved by complex step and finite differencing for a representative problem. Table III.A.1 shows some representative calculated sensitivities for the scalable problem and the agreement between the original and semi-analytic results. In this case, the complex step sensitivities calculated over the whole problem were used as the exact reference. With a sufficiently small step size, i.e the $10^{-20}$ $\pi$MDO uses as a default, these can be considered accurate to machine zero[13] and are equivalent to those obtained by automatic differentiation.[15] The semi-analytic results were found to be accurate to within $10^{-14}$ when the complex step method was used to calculate the partial derivatives and $10^{-7}$ when finite differencing was used. This greater error in the finite difference results is most likely the result of less accurate partial derivatives due to subtractive cancellation. For this particular problem, the residual based (GSE1) and residual free (GSE2) methods had the same order of accuracy but this may vary depending on problem implementation and the underlying residual codes.

| | Original | Semi-Analytic | |
|---|---|---|---|
| Sensitivity | Complex Step | Complex Step | Finite Difference |
| $\frac{df}{dz_1}$ | 1.5011619793190241 | 1.5011619793190*372* | 1.501161*8970987384* |
| $\frac{df}{dx_1}$ | 8.8643743255674643 | 8.8643743255674*483* | 8.864374*5004156376* |
| $\frac{df}{dx_2}$ | 8.3048445951781069 | 8.3048445951781*247* | 8.30484*52336145004* |
| $\frac{dc_2}{dz_1}$ | 0.0346907368144533 | 0.034690736814453*5* | 0.0346907368*695189* |
| $\frac{dc_2}{dx_1}$ | -0.0693803453125928 | -0.069380345312592*5* | -0.069380345*6558495* |
| $\frac{dc_2}{dx_2}$ | 0.0526057938095800 | 0.052605793809580*3* | 0.052605793*7144702* |
| Max Difference | - | $2.1e-015$ | $7.7e-008$ |

**Table 1. Sensitivity Validation: Selected Semi-Analytic Sensitivities calculated for the Scalable Problem**

One interesting note is that the accuracy of the final sensitivities for the MDF architecture is heavily dependent on the solution of the multidisciplinary solution (MDA) at each iteration. If the individual discipline analyses or residual codes are not converged tightly enough, the final calculated sensitivities are significantly affected. To counter this effect, a residual convergence check was added to the gauss seidel solver to ensure individual residuals are less than $1E-14$ before the MDA exits. It is left to the practitioner to adjust this convergence tolerance and trade the accuracy of final sensitivities against the computational cost of more MDA iterations. Overall, the effect of the tolerance is relatively inelastic so the default setting is $10^{-14}$for more accurate sensitivities.

The relative performance of the direct and adjoint methods are problem dependant and directly related to the number of design variables and constraints of the input problem. A study was performed to measure the performance of the adjoint and direct methods for different ratios of design variables to constraints. As $\pi$MDO is presently set up to handle only single objective optimization problems, the adjoint method will clearly be the preferred choice for the objective sensitivities. Depending on the number of constraint calculations and input design variables, the faster method, whether adjoint or direct, is automatically selected for the constraint sensitivities. This can also be determined manually by the user. A series of trials was run on a scalable mathematical problem with varying numbers of constraints and design variables. The results were calculated using the GSE2 and Adjoint2 methods (residual free), and are displayed in figure5.

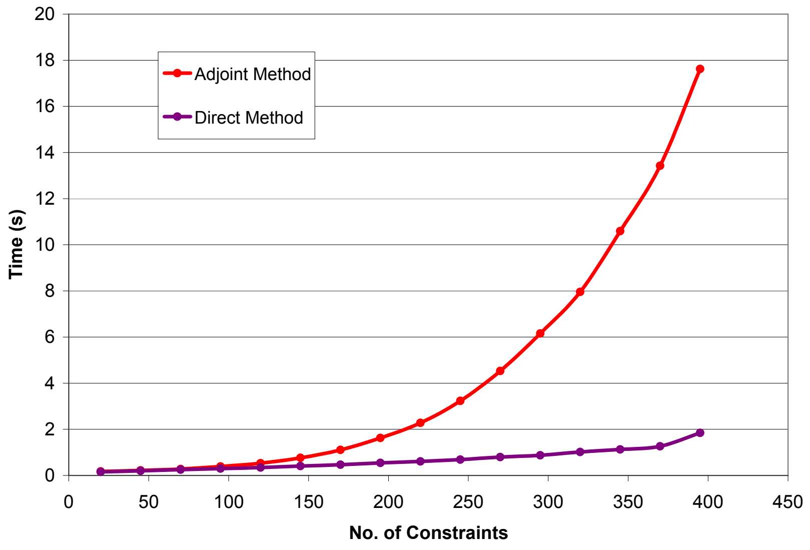

**Figure 5. Calculation times for constraint sensitivities using the adjoint and direct methods.**

### *III.A.2. Performance compared to other architectures*

The performance of the semi-analytic sensitivity methods was extensively tested and compared against a number of different MDO architectures. An initial comparison was made to the sensitivity calculation times using finite differencing. This is shown in figure 6 for one instance of the scalable problem, with three disciplines and twenty local design variables and twenty coupling variables per discipline. Whether the partial derivatives are gathered with the finite difference or complex step method, the improvement is dramatic. The majority of the improvement is due to a reduction in the number of multidisciplinary analyses (MDAs) that must be converged for each iteration. With finite difference or complex step sensitivities, one MDA is performed for each design variable at every iteration. With the semi-analytic methods, despite the higher computational overhead of the matric multiplication, only one MDA is required per iteration. This results in vast reduction in computational time and function calls.

This effect can also be shown by comparing the function calls required to converge two problems to within $10^{-6}$, shown in tableIII.A.2. The first problem is a simple two discipline analytic test problem[?],[14] and the second is a scalable test problem that can be configured with variable numbers of disciplines, local and global design variables, and coupling variables.[14] It has a quadratic objective function and discipline analysis routines, with linear constraints.

Note that the original finite difference method calls each discipline the same number of times, regardless of the dimensionality of the disciplines local and coupling variables. The partial derivative routines for the adjoint and direct methods are decoupled from each other, and hence call individual disciplines the

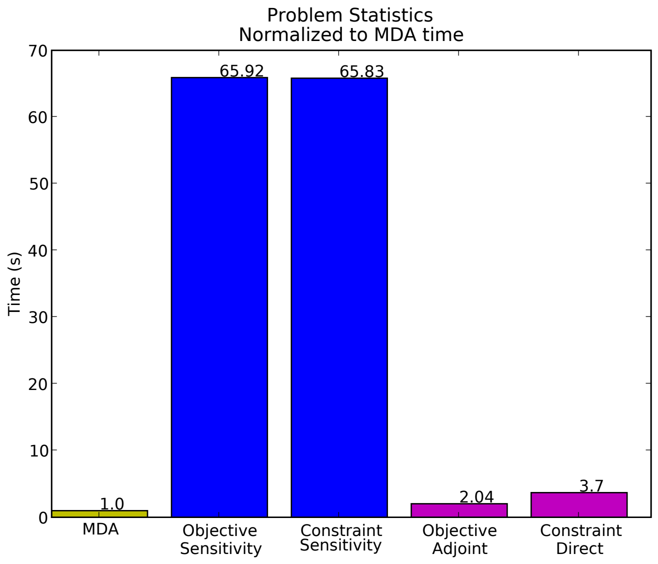

**Figure 6. Comparison of sensitivity calculation times using the original finite difference method and the new semi-analytic methods. Calculation times are normalized to the multi-disciplinary analysis performed by the MDF architecture.**

| Case |  | Function Calls |  |  |  |  |
|---|---|---|---|---|---|---|
| Architecture | Sens. Type | Discipline 1 | Discipline 2 | Discipline 3 | Discipline 4 | Total |
|  |  | **Analytic Problem** |  |  |  |  |
| MDF Original | CS | 395 | 395 | - | - | **790** |
|  | FD | 675 | 675 | - | - | **1350** |
| MDF semi-analytic | CS | 258 | 240 | - | - | **498** |
|  | FD | 313 | 277 | - | - | **590** |
|  |  | **Scalable Problem** |  |  |  |  |
| MDF Original | CS | 347638 | 347638 | 347638 | 347638 | **1390552** |
|  | FD | 380588 | 380588 | 380588 | 380588 | **1522352** |
| MDF semi-analytic | CS | 12571 | 11641 | 13501 | 12571 | **50284** |
|  | FD | 12941 | 12011 | 13871 | 12941 | **51764** |

**Table 2. Function call comparison for semi-analytic sensitivity methods. Note that this comparison is of the total analysis calls per discipline, including finite differencing.**

minimum number of times based on their complexity. This can further reduce sensitivity analysis times for problems with variable coupling between disciplines. For both these problems the discipline analysis routines are very inexpensive analytic functions. From the reduction in function calls, it is clear that the semi-analytic methods would have even more of a speed advantage for computationally expensive iterative discipline analyses. Additionally, the finite difference runs take more function calls than complex step, but the distribution is identical. Therefore, for the remainder of the trials, the complex step method was used for all architectures.

The performance of the adjoint method was also tested for increasing numbers of local design variables. For this trial, the problem was configured with 5 global design variables, and 5 distinct disciplines with 10 coupling (state) variables each. The disciplines were fully coupled, meaning the analysis of each discipline required the coupling variables from all other disciplines. The number of local design variables per discipline was increased, and several randomly generated problems were solved using various architectures. The results of this trial are shown in figure 7.

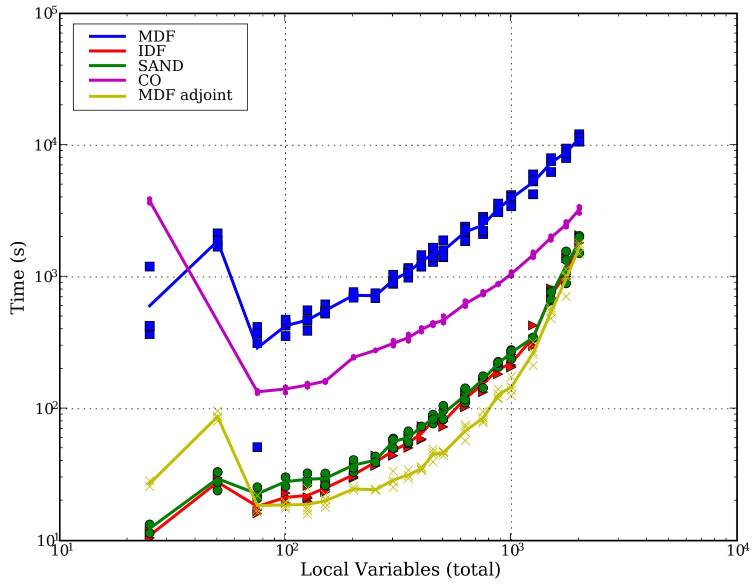

**Figure 7. Convergence times for various MDO architectures as the number of local design variables increases. As the local dimensionality goes up, the MDF method with semi-analytic sensitivities converges faster than any of the other architectures.**

As expected, the MDF method with semi-analytic sensitivities was significantly faster than the original MDF implementation using the complex step method. It was surprising how well the semi-analytic methods performed against the other architectures - as the local dimensionality increased, the modified MDF architecture converged faster than any of the other approaches. The trial was repeated, but with a random (and variable) number of coupling variables per discipline, and randomly selected coupling between disciplines. For each discipline, the number of coupling variables was randomly selected between 10 and 30 per discipline. Similarly, each discipline was coupled to 3 of the 4 other disciplines, but these connections were at random. Due to this random selection, the resulting plot is more scattered, but the overall trend is the same. The results for the randomized coupling problem are shown in figure 8.

Overall, the semi-analytic methods dramatically improved the performance of the MDF architecture. It converged faster, with fewer function calls, and more robustly than the original implementation. Further, the flexibility incorporated into the partial derivative routines allows further improvements through the use of residual codes or user provided derivatives. If the disciplines can be differentiated independantly from each other, the semi-analytic form of the MDF architecture has clear advantages for most test problems.

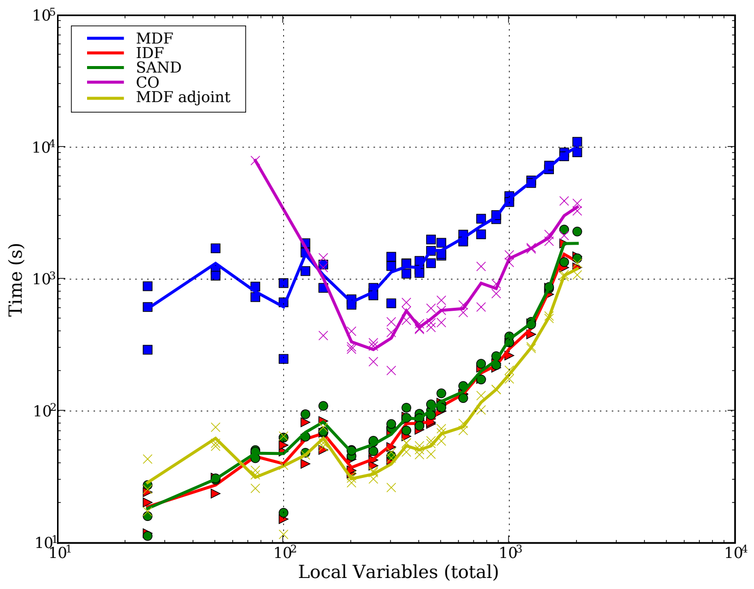

**Figure 8. Performance of various architectures for variable coupling, variable dimensionality problem. For a given number of design variables, the coupling between disciplines and number of coupling variables per discipline were randomly selected.**

## III.B. NEST Results

A version of the scalable problem was tailored to explore the advantages of the nested optimization architecture. This problem has five disciplines, two of them tightly coupled with a large number of state variables. A smaller number of state variables from these two disciplines are used in the objective function, and as coupling variables with the remaining disciplines. The structure of this test problem is shown in figure 4, and the particulars are given below.

| | Discipline 1 | Discipline 2 | Discipline 3 | Discipline 4 | Discipline 5 | Objective |
|---|---|---|---|---|---|---|
| **Inputs** | y2 | y1 | y11, y4, y5 | y22, y3, y5 | y3, y4 | y11, y22, y3, y4, y5 |
| **Outputs** | y1, y11 | y2, y22 | y3 | y4 | y5 | - |

**Table 3. Structure of scalable problem tailored to showcase reconfigurability**

The function calls required for each of the MDF, CO, and NEST architectures to converge to the optimum are given below in figure III.B.

The function calls show the new architecture allocating computational effort properly to the most highly coupled disciplines. In this initial implementation, the modified MDF architecture used for the tightly coupled disciplines does not make use of the semi-analytic sensitivity methods. Once the semi-analytic methods are implemented on the sub-optimization, the performance of the NEST architecture should improve dramatically. Further, the effect of reducing the coupling dimensionality on the system level CO problem is clearly shown - the NEST architecture has fewer function calls for disciplines three, four, and five than the CO architecture. When the number of coupling variables at the system level is increased, the performance of CO and MDF will deteriorate at a faster rate than that of the nested optimization.

Though the NEST architecture is not currently competitive with the existing architectures for this design problem, it demonstrates the inherent flexibility and power of the $\pi$MDO framework. With the addition of semi-analytic sensitivities to the nested optimizations, it is likely that the NEST architecture will converge faster, and with fewer function calls, for specific classes of problems. Further, the fact that multiple instances

| Case | Function Calls | | | | | |
|---|---|---|---|---|---|---|
| Architecture | Discipline 1 | Discipline 2 | Discipline 3 | Discipline 4 | Discipline 5 | Total |
| MDF | 79543 | 79543 | 79543 | 79543 | 79543 | **397715** |
| CO2 | 6128 | 5645 | 1768 | 1513 | 1607 | **16661** |
| NEST | 197532 | 197532 | 1355 | 1304 | 1302 | **399025** |

**Table 4. Function call comparison for semi-analytic sensitivity methods. Note that this comparison is of the total analysis calls per discipline, including finite differencing.**

of MDO architectures can be instantiated, reconfigured, and converge towards the same optimum as the original problem demonstrates the concept of reorganizing architectures within $\pi$MDO is sound. As new test problems are introduced and the architectures within the framework mature, hybrid approaches can therefore be developed with the best attributes for each situation.

## IV. Future Work

The advantage of semi-analytic sensitivity methods were conclusively demonstrated by their application to the MDF architecture and subsequent testing. Further investigation is required to see if there is any benefit to incorporating these methods into the other architectures within $\pi$MDO. These other architectures, IDF, SAND, BLISS, CSSO, etc.[14] will be examined to see if they can benefit from the application of these methods. In the case of the monolithic architectures such as SAND and IDF, the coupled sensitivity analysis would likely have no advantage when calculating the objective sensitivity. Depending on whether the constraints are global or local in nature, there may be some advantage to incorporating the global sensitivity equations.

An initial investigation proved the feasibility and relative ease of implementation of nested architectures within $\pi$MDO. Currently, the disciplines to be grouped are user defined, but automatic grouping or at least suggested configurations will be incorporated into the framework. This will be accomplished by comparing problem attributes including dimensionality of coupling between disciplines and even the values of individual sensitivities in the GSE matrix. Further, the semi-analytic sensitivity methods will be incorporated into the sub-optimizations of the NEST architecture. This should dramatically improve performance and allow larger problems, with consequently greater potential advantages to the NEST approach, to be tested. To this end, it will be applied to several test problems alongside the existing architectures in $\pi$MDO. These problems will encompass representative but largely abstract mathematical problems and more realistic medium fidelity optimization problems from the engineering world. The results gained from this process will allow the performance of each architecture in various situations to be compared and add to the knowledge base for future practitioners of MDO.

## V. Conclusion

The addition of generalized semi-analytic sensitivity methods to the $\pi$MDO framework was found to greatly increase the performance and flexibility of several architectures. When applied to the MDF architecture, these methods both decoupled the computational effort between disciplines and reduced convergence time by an order of magnitude. Exploiting the object oriented nature of $\pi$MDO, an examination was performed of the benefits of hybrid architectures incorporating nested MDO architectures. The potential of these hybrid architectures to reduce coupling dimensionality at the system level was demonstrated, and with the addition of the semi-analytic sensitivity methods these architectures have the potential to speed optimization for problems with uneven coupling between disciplines.

## Acknowledgments

The authors would like to acknowledge the support of the Canadian Natural Sciences and Engineering Research Council (NSERC) and the University of Toronto.

## References

[1]N. M. Alexandrov and S. Kodiyalam. Initial results of an MDO evaluation survey. *AIAA Paper* 98-4884, 1998.

[2]N. M. Alexandrov and R. M. Lewis. Comparative properties of collaborative optimization and other approaches to MDO. In *Proceedings of the First ASMO UK / ISSMO Conference on Engineering Design Optimization*, 1999.

[3]N. M. Alexandrov and R. M. Lewis. Reconfigurability in mdo problem synthesis, part 1. *AIAA Paper* 2004-4307, Aug. 2004.

[4]N. M. Alexandrov and R. M. Lewis. Reconfigurability in mdo problem synthesis, part 2. *AIAA Paper* 2004-4308, Aug. 2004.

[5]R. D. Braun. *Collaborative Optimization: An Architecture for Large-Scale Distributed Design*. PhD thesis, Stanford University, Stanford, CA 94305, Oct. 1996.

[6]I. R. Chittick and J. R. R. A. Martins. An asymmetric suboptimization approach to aerostructural optimization. *Optimization and Engineering*, 2008. (In press).

[7]E. J. Cramer, J. E. Dennis, P. D. Frank, R. M. Lewis, and G. R. Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization*, 4(4):754–776, 1994.

[8]V. DeMiguel and W. Murray. An analysis of collaborative optimization methods. *AIAA Paper* 2000-4720, Aug. 2000.

[9]C. A. Mader. *ADJoint: An Approach for the Rapid Development of Discrete Adjoint Solvers*. PhD thesis, University of Toronto, Toronto, ON, Oct. 2007.

[10]J. R. R. A. Martins. *A Coupled-Adjoint Method for High-Fidelity Aero-Structural Optimization*. PhD thesis, Stanford University, Stanford, CA 94305, Oct. 2002.

[11]J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft*, 41(3):523–530, 2004.

[12]J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. *Optimization and Engineering*, 6(1):33–62, March 2005.

[13]J. R. R. A. Martins, I. M. Kroo, and J. J. Alonso. An automated method for sensitivity analysis using complex variables. *AIAA Paper* 2000-0689, Jan. 2000.

[14]J. R. R. A. Martins, C. Marriage, and N. P. Tedford. $\pi$MDO: An object-oriented framework for multidisciplinary design optimization. *ACM Transactions on Mathematical Software*, 2008. (Accepted).

[15]J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The connection between the complex-step derivative approximation and algorithmic differentiation. *AIAA Paper* 2001-0921, Jan. 2001.

[16]J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, 2003.

[17]R. E. Perez. *A Multidisciplinary Optimization Framework for Flight Dynamics and Control Integration in Aircraft Design*. PhD thesis, University of Toronto, Toronto, ON, 2006.

[18]J. Sobieszczanski-Sobieski. Sensitivity of complex, internally coupled systems. *AIAA Journal*, 28(1):153–160, 1990.

[19]J. Sobieszczanski-Sobieski. Optimization by decomposition: a step from heirarchic to non-heirarchic systems. Technical report, 1998.

[20]N. P. Tedford and J. R. R. A. Martins. Comparison of MDO architectures within a universal framework. *AIAA Paper* 2006-1017, May 2006.