On Alternative Problem Formulations for Multidisciplinary Design Optimization

Ervin J. Cramer *Paul D. Frank*
Gregory R. Shubin
J. E. Dennis, Jr. *R. M. Lewis*

CRPC-TR92289
December 1992

Center for Research on Parallel Computation
Rice University
P.O. Box 1892
Houston, TX 77251-1892

Dr. Dennis wishes to credit the grant AFOSR F49620-92-J-0203.
Dr. Lewis wishes to credit The State of Texas Geophysical Parallel Computation Project, Contract 1059.

ON ALTERNATIVE PROBLEM FORMULATIONS FOR MULTIDISCIPLINARY DESIGN OPTIMIZATION

Evin J. Cramer *

Paul D. Frank

Gregory R. Shubin

The Boeing Company

P.O. Box 24346, Mail Stop 7L-21, Seattle WA, 98124-0346

J. E. Dennis, Jr.

Robert Michael Lewis

Department of Mathematical Sciences, Rice University

P.O. Box 1892, Houston, TX 77251-1892

Abstract

In this paper we introduce a perspective on multidisciplinary design optimization (MDO) problem formulation that provides a basis for choosing among existing formulations and suggests provocative, new ones. MDO problems offer a richer spectrum of possibilities for problem formulation than do single discipline design optimization problems, or multidisciplinary analysis problems. This is because the variables and the equations that characterize the MDO problem can be "partitioned" in some interesting ways between what we traditionally think of as the "analysis code(s)" and the "optimization code." An MDO approach can be characterized by what part of the overall computation is done in each code, how that computation is done, and what information is communicated between the codes.

The key issue in the three fundamental approaches to MDO formulation that we discuss is the kind of feasibility that is maintained at each optimization iteration. In the most familiar "multidisciplinary feasible" approach, the multidisciplinary analysis problem is solved at each optimization iteration. At the other end of the spectrum is the "all-at-once" approach where feasibility happens only at optimization convergence. Between these extremes lie other, new, possibilities that amount to maintaining feasibility of the individual analysis disciplines at each optimization iteration, while allowing the optimizer to drive the computation toward multidisciplinary feasibility as convergence is approached.

There are further considerations completing the classification such as how the optimization is actually done, how sensitivity information is computed from each discipline, and how, if necessary, individual gradients are combined to obtain overall problem gradients. This view of MDO problem formulation highlights the trade-offs between reuse of existing software, computing resources required, and probability of success.

1. Introduction

In this paper we examine alternative ways to formulate multidisciplinary design optimization (MDO) problems. By MDO we mean coupling together two or more analysis disciplines with numerical optimization. Due to the extreme complexity of most MDO problems, creating an MDO capability is ordinarily a difficult task requiring substantial resources. It is therefore important to understand the alternatives one has for formulating MDO problems, and the consequences of each choice.

1.1 Example problem

In thinking about MDO it is useful to have a specific problem in mind: for us it is aeroelastic optimization. We use this example to define some terms, and throughout the text to illustrate the various problem formulations. However, the formulations discussed in this paper apply to general MDO problems.

In (static) aeroelasticity we consider a flexible wing of an aircraft in steady flight. The air rushing over the wing causes pressures to be imposed on the wing, which causes the wing to deflect and change shape. This change in wing shape in turn causes the aerodynamic pressures to change. It is assumed that this process reaches an equilibrium. The two *analysis disciplines* involved are aerodynamics and structures. The computational problems

* Senior Member AIAA

for these disciplines are generally solved by individual *analysis codes*, say a finite difference computational fluid dynamics code for aerodynamics, and a finite element code for structures. The aerodynamics code takes as input the wing shape, and produces as output the pressures (and velocities, etc.) on the wing surface. The structures code takes as input the forces on the wing and a description of its structure, and produces as output the deflections (and stresses, etc.) of the wing. We say that we have *individual discipline feasibility* for aerodynamics when the CFD code has successfully solved for the pressures, given the input shape. Thus, the term "feasibility" denotes satisfaction of the equations the analysis code is intended to solve. Similarly, we have individual discipline feasibility for structures when the structures code has successfully solved the structural analysis equations to produce deflections, given some input forces.

Continuing with the aeroelastic example, we note that the two analysis codes solve their problems on different grids and interact only at a specific interface. Clearly, provision must be made for converting values of pressures from aerodynamics to forces for structures, and converting deflections from structures into changes in aerodynamic shape. We call these methods for translating between disciplines *interdisciplinary mappings*; they represent the coupling between disciplines, and play a very important role in MDO. A *multidisciplinary analysis* is achieved when we have individual discipline feasibility in aerodynamics and in structures, and the input to each corresponds to the output of the other via the interdisciplinary mappings. We call this situation *multidisciplinary feasibility*. It is possible to have individual discipline feasibility in both aerodynamics and structures, and not have multidisciplinary feasibility; this occurs if the equations in each code are satisfied, but the input to one discipline does not correspond to the output of the other. This key observation plays an important role in the "individual discipline feasible" formulation to MDO, presented later.

We next add optimization to the aeroelastic example. If we had only a single analysis discipline, aerodynamics, and we combined it with optimization, we could do aerodynamic optimization. The *design variables* would typically be some parameters, say spline coefficients, defining the wing's shape. The *objective function* would be to minimize drag, or to come as close as possible to some specified pressure distribution. There may be *design constraints* to prohibit undesirable wing shapes or bad aerodynamic flows. Similarly, with structures and optimization we could do structural optimization to minimize the structural weight by changing the size of structural components, subject to stress constraints. In aeroelastic optimization, we will generally have both *aerodynamic design variables* (shapes) and *structural design variables* (sizes, and perhaps shapes). The objective function would ideally be some measure of aeroelastic performance, but there seems to be no generally accepted single measure available in the literature. Some logical choices for the aeroelastic optimization problem are to minimize weight, subject to the constraint that drag be acceptably small, or to minimize drag, subject to weight being acceptably small. Ultimately, however, the aeroelastic behavior of the aircraft needs to be tied to some overall aircraft performance measure, like direct operating cost.

1.2 Key issues

The key issue in the alternative formulations that we present is what kind of feasibility is maintained at each optimization iteration. In the "multidisciplinary feasible" (MDF) approach, complete multidisciplinary analysis problem feasibility is maintained. In the "individual discipline feasible" (IDF) approach, individual discipline feasibility is maintained. In the "all-at-once" (AAO) approach, all of the analysis variables are optimization variables and all of the analysis discipline equations are optimization constraints. Thus, feasibility in AAO happens only at optimization convergence. (We could refer to "all-at-once" as "no discipline feasible," but we feel that "all-at-once" better describes the formulation.) In all formulations, the set of optimization variables includes the design variables. Some of the formulations induce additional optimization variables as part of their definitions.

The formulations presented here could equally well be explained from a purely mathematical point of view as specific partitions of variables and equations in the solution of the "all-at-once" optimization problem. We prefer to explain these formulations from the "discipline" point of view, because in MDO it is the disciplines that provide a natural partition of the equations and variables. However, given this link between disciplines and partitions, it is clear that all of the formulations presented here apply equally well to other notions of "discipline." For example, two disciplines could both come from aerodynamics, but one could be inviscid flow governed by the Euler equations, and the other viscous flow governed by the boundary layer equations. Alternatively, the "disciplines" could be smaller pieces of a single discipline obtained, for example, by domain decomposition [20]. Clearly, this notion can be iterated to get partitions within partitions.

We now outline the remainder of the paper. In Section 2, we present the three basic formulations for MDO problems. In Section 3, we summarize some important, but lower level, considerations that go into choosing a formulation. In Section 4, we present our conclusions.

2. Basic Approaches to MDO Problem Formulation.

We begin this section with a summary of notation and conventions. The reader may want to skip this material, and refer back to it as required. We then look at interdisciplinary mappings, and introduce the three main MDO formulation approaches: all-at-once (AAO), multidisciplinary feasible (MDF), and individual discipline feasible (IDF).

2.1 Notation, definitions, and conventions

Conventions

The notational convention used here is that X denotes variables controlled by the optimizer, C represents optimization constraints, and U denotes the analysis discipline variables computed by solving the set of analysis discipline equations W. We think of each of these as a column vector. A generic single discipline will be denoted as discipline i. For example, U_i are the analysis variables for discipline i. We use the notation $\partial C/\partial X$ to represent the Jacobian matrix of C with respect to X. Thus, $[\partial C/\partial X]_{rs} = \partial C_r/\partial X_s$ and row r is the transpose of the gradient vector of constraint r. Subscripts are used to provide more specific information about various quantities. We have ignored fixed parameters that are inputs to the analysis disciplines.

Analysis parameters

$U_i \triangleq$ Quantities internally solved for when computing an analysis for discipline i. These could include pressures, velocities, stresses, etc.

$Y_i \triangleq$ Inputs to analysis discipline i. A generic term that could include X_D, X_{ij} (see below). In particular, Y_{ij} are inputs to analysis discipline i from analysis discipline j.

$n_{A_i} \triangleq$ Total number of unknown analysis quantities, such as pressure, stresses, etc., associated with discipline i. For example, an analysis code A_i solves n_{A_i} equations for n_{A_i} analysis unknowns.

Analysis equations and mappings

$W_i \triangleq$ Equations solved in discipline i to compute the unknown analysis variables U_i. These equations can take the form $W_i(X_D, X_{ij}; U_i) = 0$. Here and throughout the report, the variables to the left of the semicolon represent inputs to the solution, while those to the right are the outputs (the variables solved for). There are usually $n_{U_i} = n_{A_i}$ of these equations.

$G_{ij} \triangleq$ Mapping to the inputs required for discipline i from the analysis variables for discipline j. For example, G_{ij} could be the mapping of the pressures on the aerodynamic grid to the loads on the structures grid. (If all the pressures from aerodynamics were directly input to structures then G_{ij} would be the identity mapping.) The functional description of this mapping is $G_{ij}(U_j)$ and there are n_{ij} of them. Most likely G_{ij} is really the composition of two functions $E_{ij}(F_j(U_j))$ that are described next.

$F_j \triangleq$ Mapping from the analysis variables for discipline j to some outputs of discipline j. For example, F_j could map the pressures computed by aerodynamic analysis to the coefficients of a spline surface approximation to the pressures. (The F is mnemonic for "fit.") (If all the pressures from aerodynamics were directly input to structures then F_j would be the identity mapping.) This approximating surface would most likely have to be evaluated before it could be input to a structural analysis code. The functional description of this mapping is $F_j(U_j)$ and there are n_j of them.

$E_{ij} \triangleq$ Mapping to the inputs required for discipline i from the fit of the analysis variables for discipline j. For example, E_{ij} could be the evaluator of a spline surface approximation to the pressures computed by aerodynamic analysis. (The E is mnemonic for "evaluate.") The evaluation of the approximating surface could then be input to a structural analysis code. (If all the pressures from aerodynamics were directly input to structures then E_{ij} would be the identity mapping.) The functional description of this mapping is $E_{ij}(F_j(U_j))$ and there are n_{ij} of them.

Optimization variables

$X_D \triangleq$ Original problem design variables. These could include wing shape parameters, beam thicknesses, etc. There are n_D original problem design variables.

$X_{ij} \triangleq$ The optimizer's estimate of parameters, required as inputs by discipline i, whose "true" values depends on the analysis solution parameters for discipline j. They look just like design variables to discipline i. (The importance of this definition is discussed in the IDF method section.) There are n_{ij} of these optimization variables arising from *coupling* estimates.

Optimization objective and constraints

$f \triangleq$ Design objective function to be minimized. This could be deviation from desired pressure distribution, drag, weight, etc. In general, f depends on the design variables X_D and the outputs U_i of all the analysis disciplines.

$C_D \triangleq$ Original problem design constraints. These could include required lift, maximum allowable stresses,

maximum wing length, etc. In general, the original problem constraints depend on the design variables X_D and the outputs U_i of all the analysis disciplines. There are m_D original problem constraints.

$C_{ij} \triangleq$ Coupling constraints among the disciplines. These constraints ensure that the optimizer's estimates X_{ij}, for the parameters mapped to discipline i from discipline j will eventually be equal to the values computed for these parameters, using the actual analysis results for discipline j. These constraints can take the form $C_{ij}(X_{ij}, U_j) \equiv X_{ij} - G_{ij}(U_j) = 0$. There are n_{ij} of these constraints. They only apply to the IDF approach.

2.2 Multidisciplinary analysis and interdisciplinary mappings

In single discipline analysis, given the inputs Y_1, the following system of equations is solved for the analysis unknowns U_1 :

$$W_1(Y_1; U_1) = 0 \tag{1}$$

Again, the semicolon notation indicates that those variables before the semicolon are inputs, those after are outputs (variables solved for). If W_1 is aerodynamics, then Y_1 can be shape parameters and U_1 can be pressures.

For multidisciplinary analysis the analysis equations are

$$\begin{aligned} &W_1(X_D, Y_{12}; U_1) = 0 \\ &W_2(X_D, Y_{21}; U_2) = 0 \\ \text{where} \quad &Y_{12} \equiv G_{12}(U_2) \\ &Y_{21} \equiv G_{21}(U_1). \end{aligned} \tag{2}$$

W_1 is still aerodynamics, and W_2 could represent structures with U_2 being displacements. The input Y_i to each discipline has been decomposed into design variables X_D and the inputs Y_{ij} to discipline i from disciplines j. The design variables are fixed for a multidisciplinary analysis, but vary for a multidisciplinary optimization. In the aeroelastic example, the design variables may be some parameters describing allowable aerodynamic shape changes, and some structural size or shape parameters. Note that the outputs from one discipline are related to the inputs of another discipline through an interdisciplinary mapping G.

In single discipline optimization, given Y_{12}, we minimize $f(X_D, U_1)$ subject to $W_1(X_D, Y_{12}; U_1) = 0$. We are not concerned with the form of Y_{12} because it does not really enter into the solution method. Y_{12} is treated as a fixed input vector.

On the other hand, for MDO the interdisciplinary mappings are important. First, the mappings really do exist. It is unusual for the outputs of one analysis code to be exactly the required inputs for another analysis code. Second, the mappings can be used to provide flexibility to change the dimension of the optimization problem by decreasing the amount of information communicated between disciplines. This could be called "reducing the interdisciplinary bandwidth." For example, the pressures computed by aerodynamics may be fit by a spline, and only the spline coefficients communicated. Note that there is an approximation process inherent in such compression. Finally, these mappings can be used to provide a common interface between codes. Suppose G=EF, where F denotes fit and E denotes evaluate. (More completely, $G_{ij}(U_j) = E_{ij}(F_j(U_j))$). For each structures code S_i we can build one fitter F_{A,S_i} that fits the output from the structures code to create the inputs for aerodynamic analysis. Then each aerodynamic code A_j needs one evaluator $E_{A_j,S}$ that evaluates structures inputs. The main point is that the aerodynamics code needs only one evaluator for all structures codes, not a different evaluator for every structures code, and similarly for the fitters.

2.3 General optimization formulation

When we combine optimization with multidisciplinary analysis, we need to solve problems of the generic constrained optimization form:

$$\begin{aligned} &\text{minimize} \quad f(X) \\ &\text{with respect to} \quad X \\ &\text{subject to} \quad C_D(X) \geq 0 \\ &\qquad\qquad\quad W(X) = 0, \end{aligned} \tag{3}$$

where the constraint equations $W(X) = 0$ ensure multidisciplinary feasibility.

As discussed later, we anticipate that most MDO efforts will involve "calculus-based" optimization techniques to solve this problem. Such techniques require gradients of the objective function f and the constraints C_D and W with respect to X. In the presentation of the AAO, MDF, and IDF formulations, we display the linearized constraint system. The entries of the linear system matrix (Jacobian matrix) represent those sensitivities that need to be calculated in order to get the gradients of the constraints for optimization. The same computations are required for the gradient of the objective function, with C_D replaced by f. The computation of these sensitivities is a critical issue for MDO, and is discussed in Section 3. In particular, it is important to note that some of the sensitivities are partial derivatives of equation *residuals* with respect to optimization variables, and others are partials of analysis *solutions* with respect to optimization variables. As described in Section 3, these different types of sensitivities are connected to each other through solutions of linear systems.

2.4 All-at-once (AAO) formulation

The "all-at-once" (AAO) formulation of design optimization problems has been mentioned in the literature both for aerodynamic optimization (e.g., [6], [7], [16], [21]) and for structural optimization (e.g., [10]). In [21] this approach is called the "one-shot" method, and in [10] it's called "simultaneous analysis and design." Here, the optimizer "controls" both the analysis and design variables, and the equations from the analysis disciplines appear as explicit constraints in the optimization. Mathematically speaking, this is really the problem we want to solve; in fact, it is useful to view all other problem formulations as iterative methods for solving the AAO problem. In AAO, we do not seek to obtain analysis problem feasibility in any sense (individual discipline or multidisciplinary) until optimization convergence is reached. In a way, the optimizer does not "waste" time trying to achieve feasibility when far from an optimum. It was found in [7] that, for aerodynamic optimization, if some computational difficulties can be overcome, AAO can be remarkably efficient. In AAO the analysis "code" performs a particularly simple function; it evaluates the *residuals* of the analysis equations, rather than *solving* some set of equations. Ultimately, of course, the optimization method for AAO must solve the analysis discipline equations to attain feasibility. Generally, this means that the solution method must contain all of the special techniques (especially for the difficult discipline aerodynamics) that a single discipline analysis solver contains. It is unlikely that "equality constraint satisfaction schemes" (e.g., Newton's method) present in existing, general purpose optimization codes would be equal to this task. Figure 1 shows the flow of information for the AAO formulation. Note that in this and subsequent figures, the codes that implement the interdisciplinary mappings are not explicitly shown as boxes in the information flow.

Using the notation defined in the previous sections, the AAO problem can be formulated as

$$\begin{aligned} &\text{minimize} && f(X_D, X_1, X_2) \\ &\text{with respect to} && X_D, X_1, X_2 \\ &\text{subject to} && C_D(X_D, X_1, X_2) \geq 0 \\ & && W_1(X_D, X_1, G_{12}(X_2)) = 0 \\ & && W_2(X_D, X_2, G_{21}(X_1)) = 0. \end{aligned} \tag{4}$$

The optimization variables are X_D, X_1, X_2. In particular, X_1, X_2 are optimization estimates of the analysis variables U_1, U_2. The number of optimization variables is $n_D + \sum_i n_{A_i}$. For our example aeroelastic problem the optimization variables are the shape and strength parameters we want to vary, the pressures and other aerodynamic variables on the aerodynamic grid, and the displacements and other structural variables on the structures grid. Note

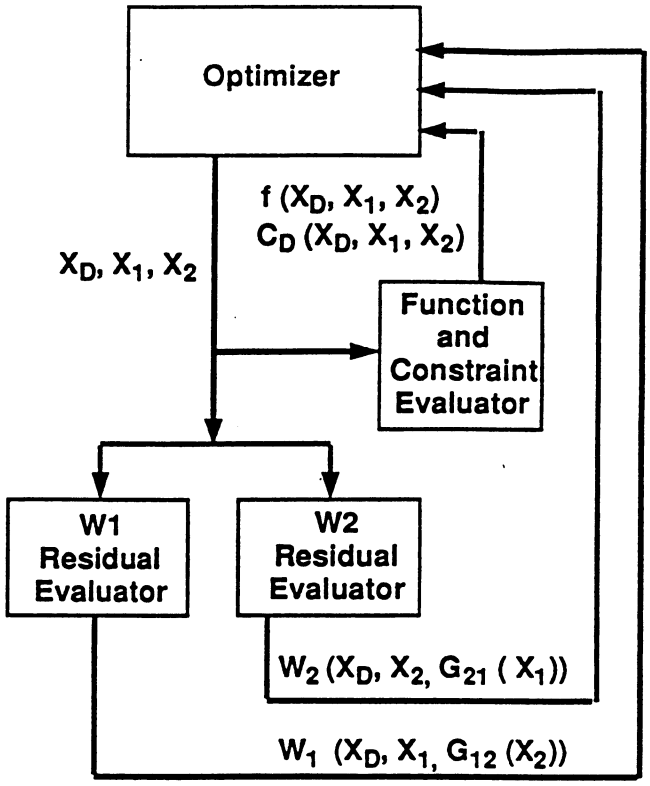

Figure 1: All-at-once method

that there is no concept of analysis codes, inputs, or outputs here; we think of a system of equations for both the analysis and the optimization that is somehow to be solved. As mentioned previously, the W_i equations are not necessarily satisfied until the optimum is found.

The linearized constraint system is

$$\begin{bmatrix} C_D^{(c)} \\ W_1^{(c)} \\ W_2^{(c)} \end{bmatrix} + \begin{bmatrix} \frac{\partial C_D}{\partial X_D} & \frac{\partial C_D}{\partial X_1} & \frac{\partial C_D}{\partial X_2} \\ \frac{\partial W_1}{\partial X_D} & \frac{\partial W_1}{\partial X_1} & \frac{\partial W_1}{\partial X_2} \\ \frac{\partial W_2}{\partial X_D} & \frac{\partial W_2}{\partial X_1} & \frac{\partial W_2}{\partial X_2} \end{bmatrix} \begin{bmatrix} \Delta X_D \\ \Delta X_1 \\ \Delta X_2 \end{bmatrix}. \tag{5}$$

where $\left(C_D^{(c)}, W_1^{(c)}, W_2^{(c)}\right)^T$ is the value of the constraints at the current point. The chain rule gives

$$\frac{\partial W_1}{\partial X_2} = \frac{\partial W_1}{\partial G_{12}} \frac{\partial G_{12}}{\partial X_2} \tag{6}$$

and a similar relation for $\partial W_2 / \partial X_1$. The terms $\partial G_{12}/\partial X_2$ and $\partial G_{21}/\partial X_1$ are derivatives of the composite processes of data-fitting and evaluation of the resulting model. It is assumed that the fitting and evaluation processes are differentiable, and that these derivatives are either analytic or inexpensive to compute.

The difficulty in computing the derivatives of the objective function f and the constraints C_D with respect to the design and analysis variables depends on the nature of the functions selected. For the AAO formulation,

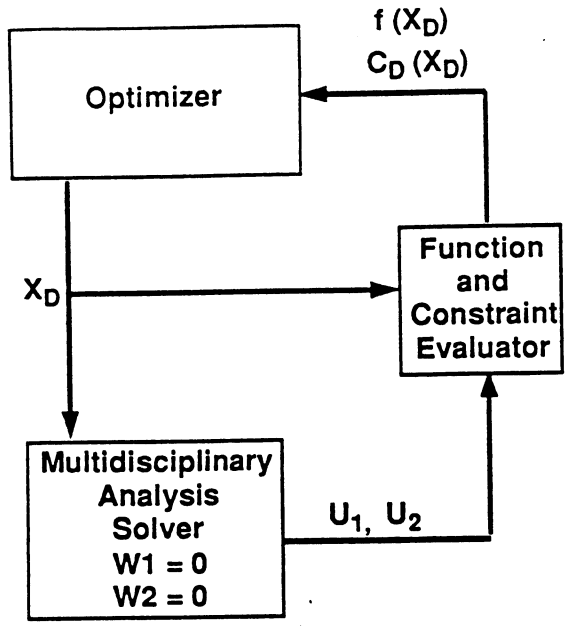

Figure 2: Multidisciplinary feasible method

it is tacitly assumed that these derivatives are easy to compute because they are derivatives of residuals or explicit functions; in other words, all variables are treated as independent quantities. The blocks $\partial W_1/\partial X_1$ and $\partial W_2/\partial X_2$ in (5) are the Jacobians that would appear in Newton solvers for disciplines one and two, respectively. The derivatives $\partial W_1/\partial X_D$, $\partial W_1/\partial G_{12}$, $\partial W_2/\partial X_D$, and $\partial W_2/\partial G_{21}$ represent the sensitivities of the analysis discipline equation residuals to their inputs. The computation of all of these sensitivities is discussed in Section 3.

2.5 Multidisciplinary feasible (MDF) formulation

As mentioned above, AAO has the disadvantage that the optimization code must assume the difficult task of simultaneously satisfying all the analysis discipline equations. The MDF formulation has the advantage that it uses the specialized software that has been developed for solving the individual discipline equations. MDF is at the opposite end of the spectrum of problem formulations from AAO. In the MDF formulation the optimizer controls only the design variables X_D, and full multidisciplinary analysis problem feasibility is maintained at every optimization iteration. In some sense, MDF is a "black-box" approach, but the black-box solves all of the analysis disciplines. Figure 2 shows the flow of information for the multidisciplinary feasible formulation.

The optimization problem is to

$$\begin{aligned} &\text{minimize} && f(X_D, U_1(X_D), U_2(X_D)) \\ &\text{with respect to} && X_D \\ &\text{subject to} && C_D(X_D, U_1(X_D), U_2(X_D)) \geq 0. \end{aligned} \tag{7}$$

To evaluate f and C_D for a given X_D, the MDF method requires solving simultaneously for U_1 and U_2 from the system of equations

$$\begin{aligned} W_1(X_D, G_{12}(U_2); U_1) &= 0 \\ W_2(X_D, G_{21}(U_1); U_2) &= 0. \end{aligned} \tag{8}$$

The optimizer only controls the variables X_D. The number of optimization variables is n_D, which is much less than in the AAO method. The optimizer no longer explicitly solves the equations for the individual disciplines. The design variables X_D may enter directly into the objective function or may indirectly affect it by changing the values of the U's.

For MDF optimization, the linearized constraint system is

$$\left[C_D^{(c)}\right] + \left[\frac{\partial C_D}{\partial X_D} + \frac{\partial C_D}{\partial U_1}\frac{\partial U_1}{\partial X_D} + \frac{\partial C_D}{\partial U_2}\frac{\partial U_2}{\partial X_D}\right]\left[\Delta X_D\right]. \tag{9}$$

where $C_D^{(c)}$ is the value of the constraints at the current point. The coefficient matrix represents the gradient of C_D with respect to the design variables X_D; the gradient of the objective function has the same form, with C_D replaced by f.

Computing the partial derivatives $\partial C_D/\partial X_D$, and $\partial C_D/\partial U_i$ is generally easy; computing the solution sensitivities $\partial U_i/\partial X_D$ is generally hard. Usually, these solution sensitivities need to be *evaluated at a multidisciplinary feasible point* during the course of optimization with the MDF formulation. Sobieski [20] has introduced two ways to combine individual discipline derivatives to obtain the required multidisciplinary sensitivities; these methods are reviewed in Section 3. Both methods exploit the analysis-discipline-based block structure of the multidisciplinary Jacobian. We observe that the MDF approach in combination with Sobieski's method for computing sensitivities can be viewed as a version of the Generalized Reduced Gradient method for optimization [14].

2.6 Individual Discipline Feasible (IDF) formulation

The MDF method has the disadvantage that a full multidisciplinary analysis is required each time the optimization code requires an objective or constraint function evaluation. By adopting the individual discipline feasible (IDF) formulation, *several* methods can be constructed that eliminate the need for multidisciplinary feasibility while taking full advantage of existing analysis codes for individual disciplines.

IDF occupies an "in-between" position on a spectrum where the AAO and MDF formulations represent extremes: for AAO, no feasibility is enforced at each optimization iteration, whereas for MDF, complete multidisciplinary feasibility is required. In between these

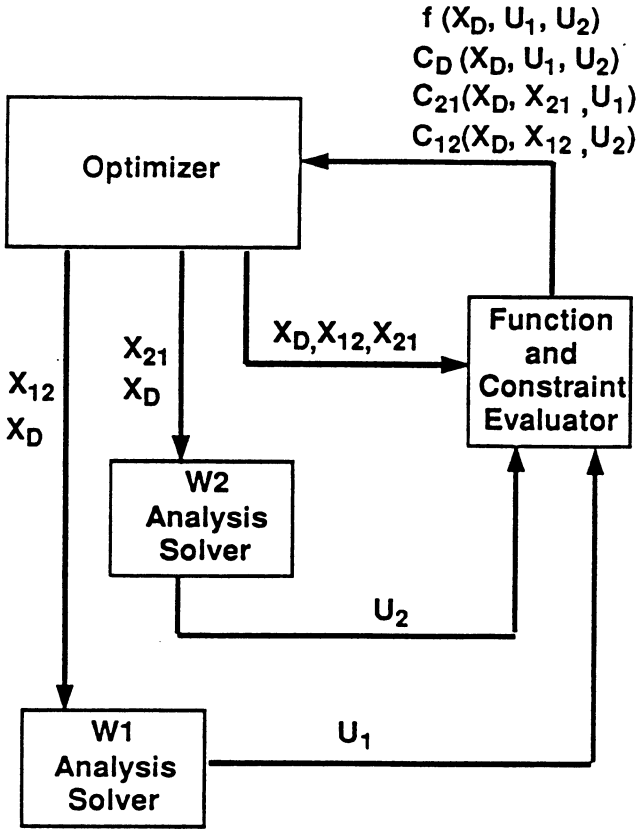

Figure 3: Individual discipline feasible method

extremes lie other possibilities that amount to specific decompositions of the work between analysis code(s) and the optimizer. The IDF approach maintains individual discipline feasibility, while allowing the optimizer to drive the individual disciplines toward multidisciplinary feasibility and optimality by controlling the interdisciplinary mappings. Since the optimizer is estimating the interdisciplinary coupling parameters, the analysis disciplines can be solved independently. The rest of this section describes one IDF method.

In an IDF aeroelastic optimization, at each optimization iteration we have a "correct" aerodynamic analysis and a "correct" structural analysis. However it is only at optimization convergence that the pressures predicted by the aerodynamic analysis will correspond to the loads input to the structures and that the displacements predicted by the structural analysis will correspond to the geometry input to the aerodynamics. Note that, in this approach, analysis variables have been "promoted" to become optimization variables; in fact, they are indistinguishable from design variables from the point of view of an individual analysis discipline solver. In IDF, the specific analysis variables that have been promoted are those that represent communication, or coupling, between analysis disciplines via interdisciplinary mappings. Figure 3 shows the flow of information for the IDF formulation.

The problem formulation for the IDF approach is

$$\begin{aligned}
&\text{minimize} \quad f(X_D, U_1(X_D, X_{12}), U_2(X_D, X_{21})) \\
&\text{with respect to } X_D, X_{12}, X_{21} \\
&\text{subject to} \\
&\quad C_D(X_D, U_1(X_D, X_{12}), U_2(X_D, X_{21})) \geq 0 \\
&\quad C_{12} \equiv X_{12} - G_{12}(U_2(X_D, X_{21})) = 0 \\
&\quad C_{21} \equiv X_{21} - G_{21}(U_1(X_D, X_{12})) = 0 .
\end{aligned} \tag{10}$$

In the above equations the composite functions G_{12} and G_{21} can be expanded as $G_{12}(U_2) \equiv E_{12}(F_2(U_2))$ and $G_{21}(U_1) \equiv E_{21}(F_1(U_1))$.

The optimization variables are now X_D, X_{12}, X_{21}. In the aeroelastic example, X_{12}, X_{21} are the inputs to aerodynamics (from structures) and to structures (from aerodynamics) on their respective grids. The optimizer is explicitly controlling some of the variables communicated between the individual disciplines. The constraints C_{12} and C_{21} have been added to assure that, at optimization convergence, the input to discipline 1 corresponds to the output of discipline 2, and vice versa. In order to evaluate the constraints and objective function in 10, we first solve for U_1, U_2 *individually* from the equations

$$\begin{aligned}
W_1(X_D, X_{12}; U_1) &= 0 \\
W_2(X_D, X_{21}; U_2) &= 0.
\end{aligned} \tag{11}$$

Note that this formulation allows us to use existing individual discipline solvers for W_i because X_D and X_{ij} are the normal inputs and U_i is the normal output of such codes. The number of optimization variables is $n_D + \sum_{i,j \neq i} n_{ij}$, which would be the same number as the AAO method if every variable from a given discipline were communicated to every other discipline. Of course, ordinarily many fewer variables are communicated.

We can take advantage of the concept of interdisciplinary mappings to define a potentially much smaller problem

$$\begin{aligned}
&\text{minimize} \\
&\quad f(X_D, U_1(X_D, E_{12}(X_{12})), U_2(X_D, E_{21}(X_{21}))) \\
&\text{with respect to } X_D, X_{12}, X_{21} \\
&\text{subject to} \\
&\quad C_D(X_D, U_1(X_D, E_{12}(X_{12})), U_2(X_D, E_{21}(X_{21}))) \geq 0 \\
&\quad C_{12} \equiv X_{12} - F_2(U_2(X_D, E_{21}(X_{21}))) = 0 \\
&\quad C_{21} \equiv X_{21} - F_1(U_1(X_D, E_{12}(X_{12}))) = 0 .
\end{aligned} \tag{12}$$

Formulation (12) differs from (10) in the following way: the optimization variables are still X_D, X_{12}, X_{21} but there are different interpretations of X_{12}, X_{21}. The optimizer is not controlling all of the analysis variables

communicated between disciplines, but is instead controlling a compressed approximation, such as the coefficients of a spline fit of the normal outputs. The number of optimization variables should be considerably smaller than for (10) that worked directly with the outputs from the codes. In order to evaluate the constraints and objective function one must solve $W_1(X_D, E_{12}(X_{12}); U_1) = 0$ for U_1 and solve $W_2(X_D, E_{21}(X_{21}); U_2) = 0$ for U_2. For this version of IDF, existing codes can still be used to solve the analysis problems. Of course, the biggest impact of reducing the number of optimization variables is the reduction in the number of sensitivities that need to be calculated.

The linearized constraint system for the IDF formulation (10) is

$$\begin{bmatrix} C_D^{(c)} \\ C_{12}^{(c)} \\ C_{21}^{(c)} \end{bmatrix} + A \begin{bmatrix} \Delta X_D \\ \Delta X_{12} \\ \Delta X_{21} \end{bmatrix} \qquad (13)$$

where

$$A = \begin{bmatrix} \frac{dC_D}{dX_D} & \frac{\partial C_D}{\partial U_1}\frac{\partial U_1}{\partial X_{12}} & \frac{\partial C_D}{\partial U_2}\frac{\partial U_2}{\partial X_{21}} \\ -\frac{\partial G_{12}}{\partial U_2}\frac{\partial U_2}{\partial X_D} & I & -\frac{\partial G_{12}}{\partial U_2}\frac{\partial U_2}{\partial X_{21}} \\ -\frac{\partial G_{21}}{\partial U_1}\frac{\partial U_1}{\partial X_D} & -\frac{\partial G_{21}}{\partial U_1}\frac{\partial U_1}{\partial X_{12}} & I \end{bmatrix}, \qquad (14)$$

$$\frac{dC_D}{dX_D} = \frac{\partial C_D}{\partial X_D} + \frac{\partial C_D}{\partial U_1}\frac{\partial U_1}{\partial X_D} + \frac{\partial C_D}{\partial U_2}\frac{\partial U_2}{X_D}, \qquad (15)$$

and $\left(C_D^{(c)}, C_{12}^{(c)}, C_{21}^{(c)}\right)^T$ is the value of the constraints at the current point. The system for formulation (12) is the same as above except that G_{12} and G_{21} are replaced by F_2 and F_1, respectively.

The expensive derivatives for the IDF method are those of the form $\partial U_i/\partial X_D$, $\partial U_i/\partial X_{ij}$, which are both sensitivities of the individual discipline analysis solutions with respect to analysis inputs. Note that the derivatives required for the IDF formulation are the same as those required by Sobieski [20] (in his GSE2 approach) for computing MDF problem derivatives. However, in contrast to the MDF method, here they only need to be *evaluated at an individual discipline feasible point.*

2.7 Alternative IDF formulations

In the IDF formulation presented above the interdisciplinary mapping (coupling) variables input into each discipline from the other disciplines were made optimization variables and the associated constraints, such as C_{12} and C_{21} in (10), were imposed. It is possible to create IDF formulations where only some of the coupling variables are optimization variables and the remainder are the actual computed analysis values. For example, the computations in the above IDF method could be sequenced such that the analysis for discipline one is completed prior to starting the analysis for discipline two. Since the inputs for discipline two from discipline one would then be available, there would be no need for the optimization variables X_{21} and the constraints C_{21}. The usefulness of such a "sequenced IDF" formulation depends on factors such as the difficulty in satisfying the coupling constraints, the cost of computing derivatives for the coupling constraints, the relative behavior of the optimization objective and constraint functions for the two formulations, and the lost opportunity for parallelism by imposing a specified sequence on the analyses.

Many different IDF formulations can be developed by using the option to sequence the individual codes. These methods are described in more detail in [3].

3. Additional Considerations in Choosing a Formulation

In Section 2, we did not discuss many important considerations that go into choosing a specific MDO formulation. Among these are choice of an optimization (or, more generally, search) technique, how to compute the required sensitivities if calculus-based optimization is used, and how simplified analyses may be substituted for more complex analyses to reduce computational expense. These issues are discussed in turn in this Section. Also there are many opportunities to exploit parallel computing in solving MDO problems; these depend on the specific formulation chosen, and are discussed in [3].

3.1 Optimization procedure

Thus far we have discussed basic approaches to MDO formulation without specifying any particular optimization algorithm to solve the problems. All of the problem formulations are constrained optimization problems and general purpose methods are available; the question is, are they applicable to solving MDO problems using these formulations of Section 2? Generally, we will choose between different optimization techniques based on problem size, smoothness, derivative availability, and sparsity.

As discussed in [5] there is no one best method for all MDO problems. Frank et al. [5] investigated the applicability of calculus-based methods (usual nonlinear programming techniques), response surfaces, expert systems, genetic algorithms, simulated annealing, and neural networks to MDO problems. They concluded that, for problems where calculus-based methods can be applied, these methods are much more effective than the other techniques. Thus, they recommended that, whenever possible, MDO problems be posed as smooth dif-

ferentiable problems so that calculus-based methods can be applied. However, they recognized that this cannot always be done. In cases where integer or discrete variables or nonsmooth functions are unavoidable, one of the search methods they discuss is appropriate. Global optimization is another case requiring use of search methods. However, even in this case, use of a calculus-based method to solve local optimization subproblems was recommended. In many situations, a combination of methods will be required.

Given a decision to use calculus-based optimization, the crucial issue becomes the efficient computation of gradients or sensitivities. This is discussed next.

3.2 Obtaining sensitivity information

There are two (related) cases that arise in sensitivity calculations: we either need the partial derivatives of the *equation residuals* with respect to the optimization variables, or we need the partial derivatives of the *analysis solutions* with respect to the optimization variables. As can be seen from the Jacobians in (5), (9), (14), the residual question pertains directly to the AAO method, while the derivatives of the analysis solution are required for the MDF and IDF methods. The derivatives of the residuals are also needed for MDF and IDF methods *if* the solution derivatives are computed via the implicit function theorem by a method described later in this Section.

Derivatives of equation residuals.

We consider the simpler case of the derivatives of equation residuals first; these are the only ones required for AAO. We then build on the residual derivatives to discuss the derivatives of the analysis solution.

Consider obtaining the Jacobian (matrix of partial derivatives) of the residuals W_1 with respect to the design variables X_D for the single discipline defined by

$$W_1(X_D; U_1) = 0. \tag{16}$$

There are three methods that we know of for obtaining the required derivatives: analytic derivation, finite difference approximation, and automatic differentiation. We briefly consider each of these possibilities in turn.

Given the equations W_1 we could certainly obtain the required partial derivatives by differentiating the expressions by hand or by symbolic manipulation, and then coding the resulting formulas. With symbolic manipulation, the code could even be generated automatically. This is a reasonable way to go when developing new code, but might be tedious and error prone when retrofitting existing code to compute derivatives.

The finite difference method is by far the most commonly used approach. Suppose that we are using one sided differences. Then, we evaluate W_1 at the nominal value of X_D and at a perturbed value of X_D, and then compute a difference approximation to $\partial W_1/\partial X_D$. For more accuracy but double the cost, we could use centered differences. In either case, we may perturb only one component of X_D at a time, or if certain components of X_D only affect certain components of W_1, we define index sets and use sparse differencing [4], [12] to reduce the number of residual evaluations required. In addition to the large cost, choosing the perturbation step size is a significant problem in the finite difference approach.

The third method for obtaining the partial derivatives, automatic differentiation, is relatively new. Automatic differentiation (see [8] for a survey of this subject) is basically a technique whereby computer code for calculating function values can be automatically augmented with code for evaluating any specified derivatives of the functions. To our knowledge, the most promising tool for making automatic differentiation a practical reality for use in MDO is ADIFOR [1].

Derivatives of analysis solutions.

We now turn to the more difficult case of obtaining partial derivatives of the analysis solution with respect to optimization variables. Consider the single analysis discipline

$$W_1(X_D; U_1(X_D)) = 0 \tag{17}$$

where we want to compute $\partial U_1/\partial X_D$. For the IDF formulation, these individual discipline sensitivities are all we need (in addition to some easily derived partials, that could be obtained by any of the residual derivative methods). The method of analytic derivation no longer applies, since we don't have explicit functional forms for $U_1(X_D)$. The finite difference method still applies, and is the most popular method in current use. Unfortunately, with this method we need to *re-solve* (17) for each perturbation of X_D to get a perturbed value of U_1, which is generally a very expensive proposition. The method of automatic differentiation also still applies, at least in principle. Here the idea is, given a code that takes as input X_D and produces as output U_1, we "simply" automatically differentiate the entire code to produce code for evaluating $\partial U_1/\partial X_D$. The authors of [15] refer to this being done on some moderately complex two dimensional aerodynamics codes, but whether it will prove practically feasible in general remains to be determined.

A very efficient way to compute $\partial U_1/\partial X_D$ for (17) is obtained by invoking the implicit function theorem. In previous papers, we have called the gradients so obtained "implicit gradients" or "cheap gradients," the latter term indicating that they are usually far less expensive to compute than finite difference approximations. The mathematical details of getting the implicit gradient formula

are given in [7], but we can obtain the same result by formally differentiating (17), at a solution of (17), with respect to X_D to obtain

$$\frac{\partial W_1}{\partial U_1}\frac{\partial U_1}{\partial X_D} = -\frac{\partial W_1}{\partial X_D} \tag{18}$$

Equation (18) is a linear system to be solved for the sensitivities $\partial U_1/\partial X_D$. The coefficient matrix $\partial W_1/\partial U_1$ may already be available in the analysis code that solves $W_1 = 0$, or it could be obtained by one of the three methods mentioned above for computation of derivatives of residuals. Similarly, the right hand side $\partial W_1/\partial X_D$ can be obtained by one of the three techniques. However, as discussed in [18] for the discipline of aerodynamics, the computation of these Jacobians can be complicated by the need to create new computational grids when design variables are changed that affect the shape of the computational domain.

As an aside, we note that there exist methods of a significantly different flavor from those discussed above for computing single discipline sensitivities. These methods appeal to the continuous (differential equations) model from which the discrete analysis solvers are derived, and apply variational calculus to the continuous problem prior to discretization. For aerodynamics, such methods have been pursued in [13], [2]. For structures, they are described in Chapter 8 of [10]. There is a strong relationship between these variational methods and the implicit gradient method discussed above; this relationship is explored in [19] for the case of aerodynamics.

At this point, we have everything we need for the IDF method.

We next examine how the methods described above may be applied to a multidisciplinary analysis to obtain sensitivities for MDF. In the same sense that they apply to individual disciplines, the finite difference and automatic differentiation methods apply to multidisciplinary analysis as well. The interesting case is the implicit gradient method. Consider the two disciplines

$$\begin{aligned} W_1(X_D, U_2(X_D); U_1(X_D)) &= 0 \\ W_2(X_D, U_1(X_D); U_2(X_D)) &= 0. \end{aligned} \tag{19}$$

Here, for simplicity, we have taken the interdisciplinary mappings G_{12} and G_{21} to be equal to the identity. Invoking the implicit function theorem, we formally differentiate (19) to obtain

$$\begin{bmatrix} \frac{\partial W_1}{\partial U_1} & \frac{\partial W_1}{\partial U_2} \\ \frac{\partial W_2}{\partial U_1} & \frac{\partial W_2}{\partial U_2} \end{bmatrix} \begin{bmatrix} \frac{\partial U_1}{\partial X_D} \\ \frac{\partial U_2}{\partial X_D} \end{bmatrix} = \begin{bmatrix} -\frac{\partial W_1}{\partial X_D} \\ -\frac{\partial W_2}{\partial X_D} \end{bmatrix}. \tag{20}$$

This is Sobieski's [20] method GSE1 for obtaining the "global sensitivities" $\partial U_1/\partial X_D$ and $\partial U_2/\partial X_D$.

In Sobieski's GSE2 equations the individual discipline solutions are explicit functions of the inputs. The following describes the computation of the "global sensitivities" $\partial U_1/\partial X_D$ and $\partial U_2/\partial X_D$ for the GSE2 case. Suppose that for any X_D and any $\bar{U}_2$ a feasible solution for discipline one is given by

$$U_1 \equiv \mathcal{F}_1(X_D, \bar{U}_2), \tag{21}$$

and similarly

$$U_2 \equiv \mathcal{F}_2(X_D, \bar{U}_1) \tag{22}$$

for any $\bar{U}_1$. Using (21) and (22), a multidisciplinary feasible solution has the property that the relations

$$U_1 = \bar{U}_1 \tag{23}$$

and

$$U_2 = \bar{U}_2 \tag{24}$$

are simultaneously satisfied.

The above notion of multidisciplinary feasibility can be captured by the constraint equations

$$\begin{aligned} W_1 &= U_1 - \mathcal{F}_1(X_D, U_2) = 0 \\ W_2 &= U_2 - \mathcal{F}_2(X_D, U_1) = 0. \end{aligned} \tag{25}$$

We can differentiate (25) to obtain

$$\begin{bmatrix} I & -\frac{\partial \mathcal{F}_1}{\partial U_2} \\ -\frac{\partial \mathcal{F}_2}{\partial U_1} & I \end{bmatrix} \begin{bmatrix} \frac{\partial U_1}{\partial X_D} \\ \frac{\partial U_2}{\partial X_D} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{F}_1}{\partial X_D} \\ \frac{\partial \mathcal{F}_2}{\partial X_D} \end{bmatrix} \tag{26}$$

which is Sobieski's GSE2 equation. As for (20), the solution of this linear system gives the solution sensitivities at multidisciplinary feasibility. The elements of the coefficient matrix are the partial derivatives of individual analysis solutions with respect to their inputs from the other disciplines, while the right hand side components are partials of individual analysis solutions with respect to the design variables. As pointed out before, the derivatives required for the coefficient matrix and right hand side in GSE2 are the same derivatives (but in a different notation) as those required for the IDF method. A very important difference is that the derivatives in GSE2 must be evaluated at a multidisciplinary feasible point.

Calculation of implicit gradients

The implicit sensitivity calculations just discussed can be used in two different ways to obtain gradients of the objective function and constraints in optimization using the MDF and IDF approaches.

Consider first single discipline optimization. The gradient of the objective function $f(X_D, U_1(X_D))$ with respect to the design variables X_D is given by

$$\nabla_{X_D} f = \left(\frac{\partial f}{\partial X_D}\right)^T + \left(\frac{\partial U_1}{\partial X_D}\right)^T \left(\frac{\partial f}{\partial U_1}\right)^T. \tag{27}$$

(The gradients of constraints are obtained by replacing f with C. The transposes appear in (27) because by convention the gradient is a column vector and the partial derivatives are row vectors or Jacobian matrices.) Here, the first term on the right indicates the explicit dependence of f on X_D, and the second term indicates the implicit dependence on X_D via U_1. In the case of implicit gradients, the term $\partial U_1/\partial X_D$ is calculated from (18), giving

$$\nabla_{X_D} f = \left(\frac{\partial f}{\partial X_D}\right)^T - \left[\left(\frac{\partial W_1}{\partial U_1}\right)^{-1}\left(\frac{\partial W_1}{\partial X_D}\right)\right]^T \left(\frac{\partial f}{\partial U_1}\right)^T \tag{28}$$

The evaluation of the term in brackets requires one linear solve with the Jacobian $\partial W_1/\partial U_1$ for each of the n_D design variables. However, if gradients of constraints are required, the term in brackets in (28) can be reused. An alternative formula for the gradient can be obtained by rearranging the linear algebra in (28) to obtain

$$\nabla_{X_D} f = \left(\frac{\partial f}{\partial X_D}\right)^T - \left(\frac{\partial W_1}{\partial X_D}\right)^T \left(\left(\frac{\partial W_1}{\partial U_1}\right)^{-T}\left(\frac{\partial f}{\partial U_1}\right)^T\right). \tag{29}$$

Evaluation of (29) requires one solve with the transpose of the Jacobian $\partial W_1/\partial U_1$, independent of the number of design variables. Whether or not it is convenient to carry out such transpose solves depends on the analysis code [18]. Unfortunately, with formulation (29) the transpose solve must be carried out again for each constraint. Thus, in general, if the number of design variables exceeds the number of nonlinear constraints, (29) is preferable, otherwise (28) is preferable [11].

The above remarks apply directly to the individual discipline feasible (IDF) approach to MDO, provided that we take into account that we need the gradients with respect to both the n_D design variables and the additional $\sum n_{ij}$ optimization variables that result from the optimizer's estimates of the interdisciplinary mapping variables. Of course, there are also $\sum n_{ij}$ additional constraint gradients that we need to compute.

The two gradient formulas (28) and (29) for single discipline optimization can be readily extended to the MDF approach for multidisciplinary optimization by using (20) instead of (18) in the derivation.

3.3 Iterative refinement

Because it is often unrealistic to use high fidelity, complex models inside the optimization loop, an alternative approach is to use simplified models inside the optimization loop and then verify the results with detailed analysis. If necessary, we can update the simplified analysis model and repeat the entire process. This process of successive refinement is shown schematically in Figure 4.

The following two observations relate the MDO formulations presented in this paper to the iterative refinement process portrayed in Figure 4. First, either AAO, IDF or MDF could be used inside the optimization loop. Second, the box containing the optimization loop could be considered as one "discipline" and the detailed analysis could be considered to be a second discipline. If an optimizer or "controller" box were added to the outer loop of Figure 4 then the resulting system is a two discipline MDO problem and is particularly amenable to an IDF approach [9].

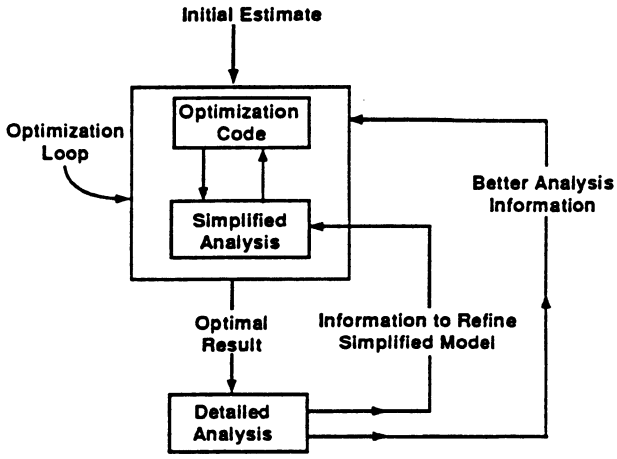

Figure 4: Successive refinement

4. Conclusions

In Table 1 we compare the features of our three main approaches to MDO formulation. In Table 2 we speculate on the performance that might be achieved by the approaches.

The multidisciplinary feasible (MDF) and individual discipline feasible (IDF) approaches have the advantage of using, with moderate or no modification, existing single discipline analysis codes. An additional advantage of IDF is that it avoids the cost of achieving full multidisciplinary feasibility at each optimization iteration, a procedure that is probably wasteful in MDF when far from optimization convergence. Furthermore, the IDF method makes it easy to replace one analysis code with another (as when additional modeling fidelity is required), or to add new disciplines.

On the other hand, the IDF approach requires the explicit imposition in the optimization of the nonlinear constraints involving the interdisciplinary maps, and the

	All-at-once (AAO)	Individual Discipline Feasibile (IDF)	Multidis ciplinary Feasible (MDF)
Use of existing codes	None	Full, no direct coupling of analysis codes	Full, but must couple the analysis codes
Discipline feasibility	None until optimal, then all disciplines feasible	Individual discipline feasibility at each optimization iteration	Multidis ciplinary feasbility at each optimization iteration
Variables the optimizer controls. (Thus, these are independent variables in sensitivities.)	Design variables and all analysis discipline unknowns	Design variables, and interdisciplinary mapping (coupling) parameters	Design variables
Number of optimization variables. (Thus, the number of sensitivites required.)	$n_D + \sum_i n_{A_i}$	$n_D + \sum_{i,j \neq i} n_{ij}$	n_D
Optimization problem size and sparsity	Very large and sparse	Moderate, size and sparsity dependent on coupling "bandwidth"	Small and dense

Table 1 Comparison of formulation features

	All-at-once (AAO)	Individual Discipline Feasibile (IDF)	Multidis ciplinary Feasible (MDF)
Probable compute time for objective and constraints	Low, evaluate residuals for all disciplines	Moderate, separately analyze each discipline	Very high, full multidisciplinary analysis
Expected overall speed of optimization process	Fast	Medium	Slow
Probability of unanalyzable intermediate designs	Low	Medium	High
Probable Robustness	???	High	Medium

Table 2 Comparison of predicted performance

calculation of additional sensitivities corresponding to the variables communicated between disciplines. If the number of such variables and constraints can be kept small, we project that the overall cost of IDF optimization will be significantly less than MDF optimization.

No matter what approach is chosen, the efficient calculation of sensitivities will be critical for success. In our opinion, with the increasing complexity of analysis codes and the increasing number of design variables that will probably be used in future MDO applications, it is unlikely that finite difference sensitivities will be affordable. In this area, the role of automatic differentiation remains to be determined. Our guess is that, for very large problems, only some kind of analytic or implicit sensitivities will be used. The other alternative, of course, is to use simplified analyses in the optimization, and correct via iterative refinement.

We feel that the all-at-once (AAO) approach remains theoretically attractive because of the probability that it will be the least expensive computationally. Unfortunately, it requires a higher degree of software integration than is likely to be achieved in the near future for realistic applications.

We reiterate that the data in Table 2 are only predictions. In a forthcoming paper, we will apply many of the methods presented here to a new model problem [17] for aeroelastic optimization.

References

[1] Bischof, C., A. Carle, G. Corliss, A. Griewank, and P. Hovland. ADIFOR, generating derivative codes from Fortran programs. Argonne Preprint MCS-P263-0991, Mathematics and Computer Science Division, Argonne National Laboratory, 1991.

[2] Cabuk, H., C.-H. Sung, and V. Modi. Adjoint operator approach to shape design for internal incompressible flows. In G. S. Dulikravich, editor, *Proceedings of the Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III)*, October, 1991.

[3] Cramer, E.J., J.E. Dennis, Jr., P.D. Frank, R.M. Lewis, and G.R. Shubin. Optimization problem formulation alternatives for multidisciplinary design. Technical Report AMS-TR-188, Boeing Computer Services, September, 1992.

[4] Curtis, A.R., Powell, M.J.D., and Reid, J.K. On the estimation of sparse Jacobian matrices. *Journal of the Institute of Mathematics Applications*, 13:117–120, 1974.

[5] Frank, P. D., A. J. Booker, T. P. Caudel, and M. J. Healy. Optimization and search methods for multidisciplinary design. AIAA-92-4827, 4th Symposium on Multidisciplinary Analysis and Optimization, September 21-23, 1992, Cleveland, OH.

[6] Frank, P.D. and G.R. Shubin. A comparison of optimization-based approaches for solving the aerodynamic design problem. Third Air Force/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, September 24-26, 1990, San Francisco, CA.

[7] Frank, P.D. and G.R. Shubin. A comparison of optimization-based approaches for a model computational aerodynamics design problem. *Journal of Computational Physics*, 98(1):74–89, 1992.

[8] Griewank, A. and G.F. Corliss, editors. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application.* Society for Industrial and Applied Mathematics, 1991.

[9] Grose, D.L. private communication.

[10] Haftka, R.T., Z. Gurdal, and M.P. Kamat. *Elements of Structural Optimization.* Kluwer Academic Publishers, 1990.

[11] Hou, G.J.-W., A.C. Taylor III, and V.M. Korivi. Discrete shape sensitivity equations for aerodynamic problems. AIAA Paper 91-2259, AIAA/SAE/ASME/ASEE 27th Joint Propulsion Conference, Sacramento, California, June 24-27, 1991.

[12] Huffman, W.P. Direct transcription and the optimal control problem: Putting the pieces together. AIAA Paper 92-4527, AIAA Guidance, Navigation, and Control Conference, Hilton Head, South Carolina, August 10-12, 1992.

[13] Jameson, A. Aerodynamic design via control theory. Technical Report 88-64, ICASE, November, 1988.

[14] Lasdon, L.S., A.D. Waren, A. Jain, and M. Ratner. Design and testing of a GRG code for nonlinear optimization. *ACM Transactions on Mathematical Software*, 4:34–50, 1979.

[15] Newman, P.A., G.J.-W. Hou, H.E. Jones, A.C. Taylor III, and V.M. Korivi. Observations on computational methodologies for use in large-scale, gradient-based, multidisciplinary design incorporating advanced CFD codes. Technical Report 104206, NASA Technical Memorandum, February 1992.

[16] Rizk, M.H. Aerodynamic optimization by simultaneously updating flow variables and design parameters. AGARD Paper No. 15, May 1989.

[17] Shubin, G.R. A new model problem for static aeroelasticity. Technical Report AMS-TR-189, Boeing Computer Services, July, 1992.

[18] Shubin, G.R. Obtaining "cheap" optimization gradients from computational aerodynamics codes. Technical Report AMS-TR-164, Boeing Computer Services, June, 1991.

[19] Shubin, G.R. and P.D. Frank. A comparison of two closely-related approaches to aerodynamic design optimization. In G. S. Dulikravich, editor, *Proceedings of the Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III)*, October, 1991.

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

[21] Ta'asan, S., G. Kuruvila, and M.D. Salas. Aerodynamic design and optimization in one shot. AIAA Paper 92-0025, 30th Aerospace Sciences Meeting, Reno, NV, January, 1992.