Problem Formulation for Multidisciplinary Optimization

Evin J. Cramer,
J. E. Dennis, Jr.,
Paul D. Frank
Robert Michael Lewis and
Gregory R. Shubin

CRPC-TR93334
August 1993

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

Problem Formulation for Multidisciplinary Optimization

Evin J. Cramer * J. E. Dennis, Jr. † Paul D. Frank *
Robert Michael Lewis † Gregory R. Shubin *

August 25, 1993

Abstract

This paper is concerned with the optimization of systems of coupled simulations. In computational engineering, this frequently is called multidisciplinary (design) optimization, or MDO.

We present an expository introduction to MDO for optimization researchers. We believe the optimization community has much to contribute to this important class of computational engineering problems. In addition, this paper presents a new abstraction for multidisciplinary analysis and design problems as well as new decomposition formulations for these problems. Furthermore, the "individual discipline feasible" (IDF) approaches suggested here make use of existing specialized simulation analysis codes, and they introduce significant opportunities for coarse-grained computational parallelism particularly well-suited to heterogeneous computing environments.

The key issue in the three fundamental approaches to MDO formulation discussed here is the kind of feasibility that must be maintained at each optimization iteration. In the most familiar "multidisciplinary feasible" (MDF) approach, the multidisciplinary analysis problem is solved multiple times at each optimization iteration, at least once everytime any problem function or constraint or derivative is evaluated. At the other end of the spectrum is the "all-at-once" (AAO) approach where feasibility is guaranteed only at optimization convergence. Between these extremes lie the new IDF formulations 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 in choosing a formulation, such as what sensitivities are required and how the optimization is actually done. Our discussion of these and other issues related to MDO problem formulation highlights the trade-offs between reuse of existing software, computational requirements, and probability of success.

Keywords: Constrained optimization, multidisciplinary design optimization, optimal design, computational engineering.

---

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

†Department of Computational and Applied Mathematics, Rice University, P.O. Box 1892, Houston, TX, 77251-1892. This work was supported by the State of Texas under contract #1059, the Air Force Office of Scientific Research under grants F49629-92-J-0203 and F49629-9310212, the Department of Energy under grant DE-FG005-86ER25017, the Army Research Office under grant DAAL03-90-G-0093, and the National Science Foundation under cooperative agreement CCR-9120008.

1 Introduction

A fundamental challenge for the emerging field of computational engineering is the design and analysis of systems and products described by coupled simulations. This field is called multidisciplinary design optimization, multidisciplinary optimization, or (most commonly) MDO. There is a tremendous economic significance to developing this capability in order to facilitate rapid prototyping of new designs and to reduce the time-to-market for new products.

Mathematical programmers have much to contribute to the problem of optimizing complex coupled systems of single discipline simulations, especially through the introduction of modern algorithms for nonlinear and integer programming to the field of MDO. The mathematical programming community may also be motivated to put more effort into the nonlinear mixed integer programming (MIP) problems that arise commonly in MDO. Nonlinear MIP is a fiendishly difficult area, but we hold some hope because, historically, the great advances in discrete optimization have come from studying particular classes of problems.

Due to the extreme complexity of most MDO problems, we believe it is necessary to focus on problem formulation methods and their interdependence with nonlinear programming algorithms. In this paper we present a new abstraction of the MDO problem, and we use it to examine alternative ways to formulate MDO problems. We do not concentrate here on the application of mathematical programming algorithms to MDO. The question of formulating MDO problems is a major topic in the engineering literature on MDO (e.g., [15]).

In part, the genesis of the ideas given here was to find ways to exploit parallel computation in nonlinear programming. We turned to reformulating the problems because the design of parallel algorithms for general nonlinear programming has not been very successful. The new IDF formulations we suggest here have the advantage of a coarse-grained parallelism naturally suited to a heterogeneous computing environment. We hasten to point out that parallelism is not the only motivation for this formulation; indeed, even without parallelism, we expect IDF to be an efficient approach for solving MDO problems.

In Section 2, we present aeroelastic optimization as an example to provide intuition into the concepts presented in the rest of the paper. In Section 3, we present notation and definitions describing our abstract model of the multidisciplinary analysis and optimization problem. This model is used in Section 4 to discuss multidisciplinary analysis, the attainment of feasibility for MDO. Section 5 is a discussion of the three main formulations for MDO including one hinted at in other work, but stated explicitly here. Section 6 discusses the derivative requirements for MDO. Section 7 discusses some issues related to choosing a formulation. In Section 8, we present our conclusions.

The contents of this paper represent an abstraction and generalization of more specific material presented in [3]. It is natural to ask whether the generalization is worthwhile, or whether this paper should have the more limited goal only of introducing optimizers to the interesting, important, and challenging problem of MDO.

We found the abstraction essential for exposition, which, in retrospect, is no surprise, since it was necessary to extend our research in order to fill gaps in the exposition. As we worked together to write the intended expository paper, we found ourselves forced again and again to refine our earlier formalization to communicate essential features of the structure of MDO problems even to one other. We have come to regard the abstraction of MDO that emerged as a useful contribution, and we find that having absorbed it, we see the MDO structure it represents everywhere we look.

2 Example - Aeroelastic Optimization

A specific problem is very useful in thinking about MDO. For us the model problem is aeroelastic optimization. We use this example to define some terms, and throughout the text we will refer to this example to illustrate the model and the various problem formulations. However, the model and formulations discussed in this paper are meant to 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. In static aeroelasticity, we assume that these physical processes reach an equilibrium.

The aeroelastic system in equilibrium is shown in Figure 1. The two *analysis disciplines* involved are aerodynamics (D_A) and structures (D_S). The computational problems for these disciplines are generally solved by individual *analysis codes*, say, a finite difference computational fluid dynamics (CFD) code for aerodynamics (A_A), and a finite element code for structures (A_S).

It is very important that the reader understand the significance of the diagrams such as Figure 1 that appear in this paper, what they convey, and what they do not convey. These diagrams are intended to show how their components are related, insofar as information is transmitted between them. In this regard, they capture a purely static feature of the problem. These diagrams are *not* intended to be flowcharts, or to express any sequence of actual calculations or operations. Indeed, the diagrams that we represent have no sources or sinks, so that they cannot function as flowcharts describing a sequence of computation with a beginning and end. Nonetheless, we shall see the utility of these diagrams in their ability to capture the structure of the coupling and communication between components in the problem.

Suppose that the structures code has been given a description of the wing structure, and that both the aerodynamics and structures code have been given a description of the undeflected wing shape. The aerodynamics code takes as an additional input the wing deflections (M_{AS}), and produces as output the pressures (and velocities, etc.) (U_A) on the wing surface. The structures code takes as an additional input the load on the wing (M_{SA}), and produces as output the deflections (and stresses, etc.) (U_S) of the wing. We say that we have *single discipline feasibility* for aerodynamics when the CFD code (A_A) has successfully solved for the pressures, given an input shape. Similarly, we have single discipline feasibility for structures when the structures code (A_S) has successfully solved the structural analysis equations to produce deflections, given some input forces. Thus, "feasibility" for a single discipline means that the equations the discipline code is intended to solve are satisfied.

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. We accommodate this by following each analysis map, e.g., (A_A), by a map (F_{SA}) that represents something like a

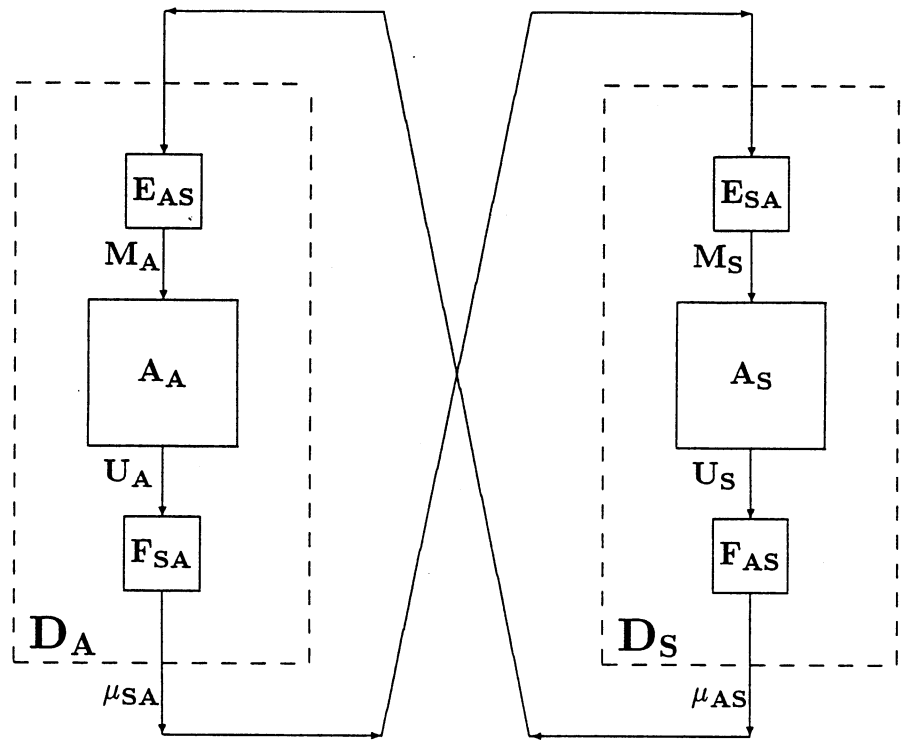

FIGURE 1: Aeroelastic System

spline fit to the grid values generated, and preceding the following analysis code by a map (E_{SA}) that represents something like a spline evaluation to generate values at points needed by that analysis code, e.g., (A_S).

But this is not enough. Clearly, provision must be made for converting values of pressures from aerodynamics into values of forces (the integral of pressure) for structures, and converting deflections from structures into changes in aerodynamic shape. We will always assume that this conversion is done either in the fit or the evaluate routine, but we do not require that the same choice be made on both sides of a given analysis code. For example, we assume that pressures are converted to loads either in F_{SA} or in E_{SA}. Independently, we assume that deflections are converted into shape changes either in F_{AS} or in E_{AS}. It will also be convenient to view a discipline as the analysis code together with all the E and F codes used to get its input and provide its output to other disciplines; this notion is depicted by the dashed boxes in Figure 1.

We call all of the maps $E \circ F \triangleq G$ *interdisciplinary mappings*. They represent the coupling between disciplines, and play a key role in MDO. We tacitly assume that each instance of an E or an F takes inputs from a single discipline and sends outputs to a single discipline. One way to handle instances of E or F that have more complex communication is to treat the subject mapping as a new "discipline". Note that the data passed between the disciplines in Figure 1 may be considered "compressed" if μ_{SA} and μ_{AS} are much smaller vectors than U_A and U_S, respectively. This would happen if, for example, the μ vectors represented coefficients of fitting functions with the U vectors as data. We call this "reducing the interdisciplinary bandwidth." Note that an approximation would be made in such a fitting operation. As shown later, this data compression can be used to reduce the dimension of the optimization problem in certain formulations. These interdisciplinary mappings could also be used to provide a common interface between codes.

A *multidisciplinary analysis* is achieved when

1. We have single discipline feasibility in aerodynamics and in structures, and

2. The input to each corresponds to the output of the other via the interdisciplinary mappings.

We call this situation *multidisciplinary feasibility* and it corresponds to the simulations in Figure 1 being in equilibrium. We discuss obtaining multidisciplinary feasibility in Section 4.

It is possible to have single discipline feasibility in both aerodynamics and structures (we call this *individual discipline feasibility*) 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 later when we present the "individual discipline feasible" (IDF) formulations for MDO.

We next add optimization to the aeroelastic example. Aerodynamic optimization combines the single analysis discipline aerodynamics with optimization. The *design variables* would typically be some parameters, say spline coefficients, defining the wing's shape. The *objective function* might 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, structural optimization combines structures and optimization to minimize the structural weight by changing the size of structural components, subject to stress constraints. In aeroelastic optimization, the combination of aerodynamics, structures and optimization, we will generally have both *aerodynamic design variables* (shapes) and *structural design variables* (sizes, and perhaps shapes). In our model, we lump all the design variables into a single variable X_D.

The objective function must be some measure of aeroelastic performance, but there seems to be no generally accepted single measure available in the aeronautical 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. Alternatively, minimizing a combination of drag and weight might be appropriate. Ultimately, however, the aeroelastic behavior of the aircraft needs to be tied to some overall aircraft performance measure, like direct operating cost. This situation, with a set of conflicting objectives, is to be expected in MDO because engineers in each discipline will probably have formulated their own objectives for the design. We will not consider the matter further in this paper, but the reader will note a goal programming approach to nonconvex multiobjective optimization in these comments.

In order to appreciate the trade-offs between the various formulations of MDO problems presented later, it is necessary to know something about the size and difficulty of the underlying analysis disciplines. Obviously these depend on the problem, but we assume that the problem is sufficiently complex that it cannot simply be overwhelmed with computing power. For example, a practical aeroelastic optimization for a three-dimensional configuration will involve a computational fluid dynamics code that takes hours of supercomputing time to execute a single analysis. The structural analysis code will typically be less costly, but may take a significant fraction of an hour. For either code the amount of computing time is acceptable for engineering analysis. However, many formulations of MDO require tens to hundreds of such executions; thus the impetus for MDO formulations requiring less computational work, and the need to employ parallel computing even for the cheaper methods. A discussion of further considerations in choosing a formulation is postponed until Section 7.

3 A Framework for Describing MDO Problems

In this section, we generalize the two discipline aeroelastic MDO example to an abstraction for reasoning about general MDO problems. This is important in its own right for the development of MDO as a research area, but it is also necessary before we can be more specific about the various optimization formulations. This rigorous summary of notation and conventions will be very useful, if somewhat tedious and delicate. The reader may want to make sure of understanding Figure 1 and refer back to the definitions as required. Figure 2 shows the data flow for a single discipline of a many-discipline version of Figure 1.

In our notational convention, X denotes the vector of variables controlled by the optimizer (nonbasic variables). The original design variables X_D are always components of X, but in some formulations, X includes other variables as well. Constraints in the optimization problem are denoted by $C(X)$. The original design or system constraints $C_D(X)$ are always components of C, but C can include other constraints.

The notation $\partial C/\partial X$ represents the Jacobian matrix of C with respect to X. Thus, $[\partial C/\partial X]_{rs} = \partial C_r/\partial X_s$ and its rth row is the transpose of the gradient vector of constraint C_r.

An important convention is the way we use subscripts. When a quantity has double subscripts, the order indicates information flow as in "to-from." For example, denote a generic ith single discipline by D_i. Then information meant to pass *to* D_i *from* D_j will be subscripted ij. It is useful to think of a discipline D_i as a grouping of communication and analysis codes. In terms of the example, the structural analysis code might have an "evaluator" code to provide loads where they are needed for the structural analysis. It might also have a "fitter" code to compress its output for communication to other disciplines. To avoid even more complexity, we allow these routines to pass some variables, like X_D, directly through.

We use the convention that arguments to the left of a semicolon are inputs to a function of a vector variable, and those to the right are the dependent variables to be determined by an equation involving the function.

In Figure 2, we assume that X_D is available to all the computations within the discipline. We assume that constants that are needed for analysis but are not mentioned in the notation, such as the Reynolds number of the flow, reside where they are needed.

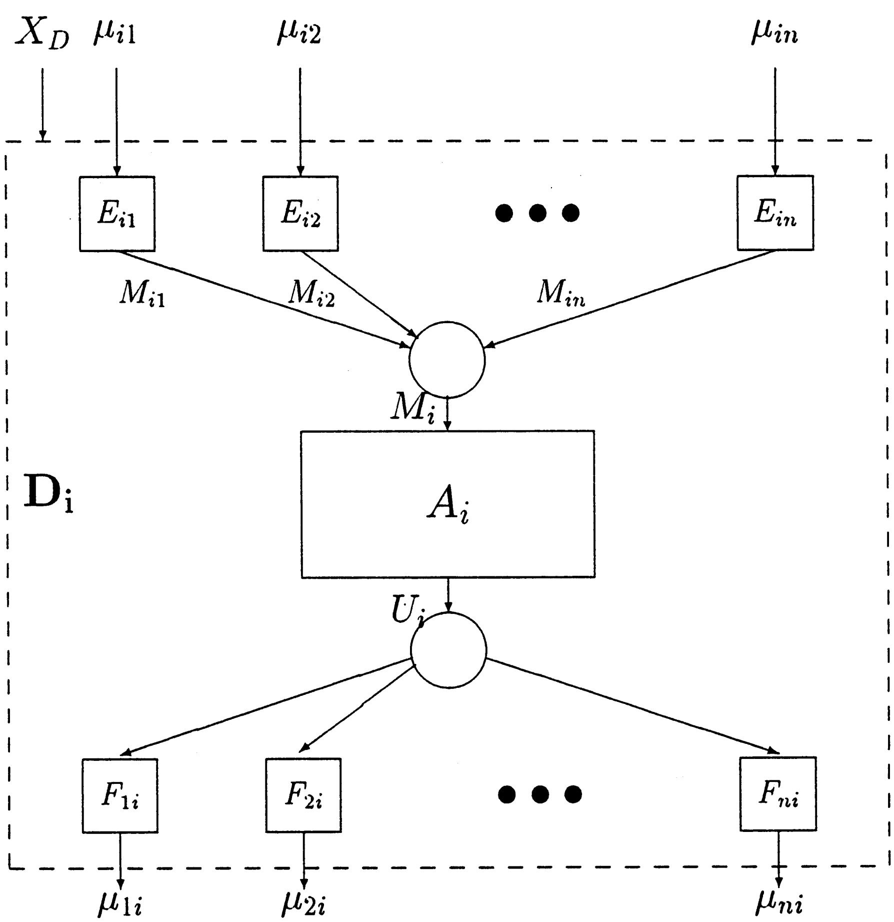

FIGURE 2: One of many disciplines

3.1 Analysis inputs, equations, and outputs

$M_i \triangleq$ Inputs to the analysis code A_i of discipline D_i. Block components M_{ij} of M_i are inputs to analysis code A_i needed from discipline D_j. The vector M is the block vector comprised of all the block vectors M_i, for every i. (The M is mnemonic for "multidisciplinary data." A_i is to be executed with M_i and design parameters X_D as inputs.)

$n_{M_i} \triangleq$ The total number of interdisciplinary inputs to A_i, i.e., the length of the vector M_i.

$A_i \triangleq$ The analysis code or solver mapping of the form $U_i = A_i(X_D, M_i)$. Much effort and talent have gone into developing these codes, and so there are serious advantages to formulations that preserve their integrity.

$U_i \triangleq$ Quantities for which A_i solves internally when executed in D_i. These could include pressures, velocities, stresses, etc. As above, U denotes the vector of analysis discipline variables (basic variables) computed in a given formulations by solving the complete set of analysis discipline equations.

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

$W_i \triangleq$ Residual function of equations solved in D_i by A_i to compute the analysis variables U_i. These equations take the form $W_i(X_D, M_i; U_i) = 0$. We remind the reader that the variables to the left of the semicolon represent inputs to the system, while those to the right are the outputs (the variables for which A_i solves). There are n_{U_i} of these residuals.

3.2 Interdisciplinary mappings

$G_{ij} \triangleq$ Mapping *to* the inputs required for analysis code A_i *from* the output of A_j. For example, G_{ij} could be the mapping of the pressures on the aerodynamic grid to the loads on the structures grid. The functional form of this mapping is $M_{ij} = G_{ij}(X_D, U_j)$, where the G_{ij} are the composition of two mappings $E_{ij} \circ F_{ij}$ given below. Sometimes there will be no input to A_i from A_j. Our convention for this is to set $G_{ij} = 0$.

$F_{ij} \triangleq$ Mapping from the analysis variables from A_j to the outputs of D_j needed to produce input to D_i. For each i, j, F_{ij} has the role of transforming U_j for use in discipline i. This transformation will often involve a data compression to reduce the communication bandwidth between disciplines. For example, $\mu_{ij} = F_{ij}(X_D, U_j)$ could map the pressures computed by aerodynamic analysis to the coefficients of a spline surface approximation to the pressures or to coefficients for a fit to the load induced on the wing by those pressures. (The F is mnemonic for "fit.") There are n_{ij} such vector functions. Some may be identity mappings, and some may be zero mappings.

$\mu_i \triangleq$ Inputs to D_i from other disciplines. Block components μ_{ij} of μ_i are sent by the fitter of D_j to the evaluator of discipline D_i. Our convention is that μ_{ij} may be just a compression of U_j by F_{ij} which will be transformed into M_{ij} by E_{ij}. Alternatively, μ_{ij} may be the product of a more complicated transmogrification. The symbol μ is mnemonic for "M or U". It is intended to reflect the nature of μ as a surrogate for M or U depending on the particular pair F_{ij}, E_{ij}. The vector μ is the block vector of all block vectors μ_i for every i.

$n_{\mu_i} \triangleq$ The total number of inputs μ_{ij} to D_i. That is, $n_{\mu_i} \equiv \sum_{j,j\neq i} n_{\mu_{ij}}$

$E_{ij} \triangleq$ Mapping to the inputs required for A_i from the compressed μ_{ij} from D_j. For example, $M_{ij} = E_{ij}(X_D, \mu_{ij})$ could be the evaluator of a spline approximating structural loads with coefficients μ_{ij}, or if μ_{ij} is the vector of coefficients of a spline fit to pressure, then the convention is that E_{ij} also performs the integration to obtain loads. (The E is mnemonic for "evaluate.") We will assume that for each F_{ij} there is a corresponding E_{ij}; some of the E_{ij} may be identity mappings, and some may be zero maps.

The reader will see that the separation between A_i and its evaluators, and the flow of information only in the direction from the evaluators to the analysis code, are likely simplifications of the true relationships of these components. For instance, if A_j is a code involving an adaptive grid, in the course of performing its analysis A_i may need to return to its evaluators to obtain information for the adaptively updated grid. This complication is not a problem if the reader bears in in mind that the purpose here is to represent the flow of information between disciplines.

For all of the preceding, A, G, F, E, and W will denote the block vector functions comprised of all of the corresponding subscripted functions, for all i, j. In order to use this compact notation, it is necessary to keep in mind that the ordering of the components must be different to be consistent. For example, suppose that D_1 is aerodynamics and D_2 is structures. Then if we order U as U_1, U_2, we must order A as A_1, A_2 and we must order G as G_{12}, G_{21} in order to have the convenience of writing $U = A(X_D, G(X_D, U))$ to express the equilibrium of the aeroelastic system.

3.3 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.

$X_\mu, X_U, X_M \triangleq$ Optimizer supplied values respectively for μ, U, M. These are not used in all formulations. In some formulations, the optimizer will explicitly control not only X_D, but also a subset of these surrogates for μ, U, M. They look just like design variables to D_i.

$X \triangleq$ The vector of all block variables X_D and any of X_μ, X_U, X_M that are explicitly controlled by the optimizer. For convenience, we will sometimes use X as surrogate arguments in a function that we have defined above in terms of the original system variables. For example, if we have specified in a particular MDO formulation that X has components X_D, X_μ, then we may write $M = E(X)$.

3.4 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 explicitly on the design variables X_D and the outputs U of all the analysis disciplines.

$C_D \triangleq$ Original problem design constraints. These could include required lift, maximum allowable stresses, maximum wing length, etc. The constraints in the original problem depend explicitly on the design variables X_D and the outputs U of all the analysis disciplines.

$C_{aux} \triangleq$ Coupling constraints among or within the disciplines, needed for formulations in which the optimizer explicitly controls more parameters than X_D. These constraints ensure that feasibility for the reformulated MDO problem is achieved at optimization convergence. For example, if the optimizer controls a surrogate X_U for U, then an auxiliary constraint like $C_{aux}(X) \equiv W(X_D, G(X), X_U) = 0$ would be needed for the MDO problem.

$C \triangleq$ The vector function of residuals of all constraints C_D and C_{aux} to be satisfied by the optimizer.

4 Feasibility

As described in Section 2, a *multidisciplinary analysis*, or MDA, is achieved when the coupled system is in equilibrium. We say that a MDA has been completed when the values of all the variables do not change upon execution of all mappings shown in Figure 1 without regard to order.

MDA can be very costly because of the expense incurred by an MDA algorithm in repeatedly executing the analysis codes. One way to avoid some of this cost is not to require feasibility until convergence to optimality. However, there will be approaches in which we will require partial feasibility for some very good reasons. The point of this section is to express the notions of feasibility needed later by using the framework provided in the previous section.

We say that a single discipline analysis has been carried out for a particular D_i when $W_i(X_D, M_i; U_i) = 0$ has been solved to yield U_i for the given inputs X_D, M_i. This would probably be done by executing A_i for the given input. When $W_i(X_D, M_i, U_i) = 0$ we have *single discipline feasibility* for discipline i. Note that we do not use the semicolon to separate the arguments into input and output for this point of view. The design variables X_D are fixed for a multidisciplinary analysis, but of course, they vary for a multidisciplinary optimization.

Likewise, using the residual form, we say that we have *individual discipline feasibility* when

$$W(X_D, M, U) = 0 \tag{1}$$

or when U has been computed in explicit form as $U \equiv A(X_D, M)$. Equation (1) states that individual discipline feasibility implies that *every* discipline has single discipline feasibility.

We have *multidisciplinary feasibility*, or MDF, when, in addition to individual discipline feasibility, the interdisciplinary variables match. In the residual form this is

$$W(X_D, M, U) = 0 \quad \text{and} \quad M = G(X_D, U) \ . \tag{2}$$

In the nonresidual form it is

$$U = A(X_D, M) \quad \text{and} \quad M = G(X_D, U) \ . \tag{3}$$

We can combine each of the residual and nonresidual forms into an equivalent equation:

$$W(X_D, G(X_D, U), U) = 0 \quad \text{or} \quad U = A(X_D, G(X_D, U)) \ . \tag{4}$$

For the aeroelastic example, the residual form of (1) is

$$\begin{aligned} W_A(X_D, M_{AS}, U_A) &= 0 \\ W_S(X_D, M_{SA}, U_S) &= 0 \end{aligned} \tag{5}$$

and the interdisciplinary constraints are

$$\begin{aligned} M_{AS} &= G_{AS}(X_D, U_S) \\ M_{SA} &= G_{SA}(X_D, U_A) \ . \end{aligned} \tag{6}$$

Thus in the residual form of MDF, we simultaneously satisfy (5) and (6). The complete nonresidual form is

$$\begin{aligned} U_A &= A_A(X_D, M_{AS}) \\ U_S &= A_S(X_D, M_{SA}) \\ M_{AS} &= G_{AS}(X_D, U_S) \\ M_{SA} &= G_{SA}(X_D, U_A) \ . \end{aligned} \tag{7}$$

The residual form of the combined equation (4) that expresses multidisciplinary feasibility in terms of just the variables X_D and U is

$$\begin{aligned} W_A(X_D, G_{AS}(X_D, U_S), U_A) &= 0 \\ W_S(X_D, G_{SA}(X_D, U_A), U_S) &= 0 \ . \end{aligned} \tag{8}$$

To reiterate, the difference between individual discipline feasibility and multidisciplinary feasibility is the matching of interdisciplinary input and output variables to reflect equilibrium. Since traditional single discipline optimization is just optimization under the constraint (1) for one discipline, this makes the point that $M = G(X_D, U)$ is the constraint that distinguishes both MDA and MDO from their single discipline counterparts.

In some applications it may be expedient always to enforce equilibrium between certain some pairs of disciplines D_i, D_j. We model this case by coalescing the pair into a single composite discipline. The term "tight coupling" is sometimes used by engineers to describe this coalescence, in which two disciplines D_i and D_j are conjoined to produce a single analysis code that simultaneously solves

$$\begin{aligned} W_i(X_D, M_i; U_i) &= 0 \\ W_j(X_D, M_j; U_j) &= 0 \\ M_{ij} &= G_{ij}(X_D, U_j) \\ M_{ji} &= G_{ji}(X_D, U_i) \ . \end{aligned} \tag{9}$$

5 MDO Formulations

Up to this point, we have used our framework for MDO to discuss various kinds of feasibility for the coupled MDA system. The purpose of this section is to widen our discussion to include optimization.

The key issue in the alternative formulations that we present here is the kind of discipline feasibility maintained at every function, constraint, or sensitivity needed during each optimization iteration. In the "multidisciplinary feasible" (MDF) approach, complete multidisciplinary analysis problem feasibility is maintained at every optimization iteration. In the "individual discipline feasible" (IDF) approach, only individual discipline feasibility (i.e., single discipline feasibility for all of the disciplines) is maintained. The interdisciplinary equilibrium constraints are added as optimization constraints in order to ensure a full MDA at optimization convergence. 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 and IDF is guaranteed 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 include additional optimization variables as part of their definitions.

In the following subsections, first we give the general mathematical specification of the MDO problem formulations and then we give the specialized aeroelastic formulations.

5.1 Formulations for general MDO problems

5.1.1 Multidisciplinary Feasible (MDF) Formulation

The most common way of posing MDO problems is, in our terminology, the multidisciplinary feasible, or MDF, formulation. In this formulation, the vector of design variables X_D is provided by the optimizer to the coupled system of analysis disciplines and a complete MDA is performed with that value of X_D to obtain the system output variable $U(X_D)$ that is used in evaluating $f(X_D, U(X_D))$ and $C_D(X_D, U(X_D))$. The MDF formulation is

$$\begin{array}{ll} \text{minimize} & f(X_D, U(X_D)) \\ \text{with respect to} & X_D \\ \text{subject to} & C_D(X_D, U(X_D)) \geq 0 \end{array} \tag{10}$$

where $U(X_D) = A(X_D, G(X_D, U(X_D)))$.

The reader may recognize MDF as a reduced basis formulation in which X_D is the nonbasic vector and everything else is a basic vector. To avoid possible conflicts with the normal usage of the terms basic and nonbasic we will say that X_D is an *explicit* variable and that U and all the other variables that arise in the MDA part of the problem are *implicit.*

The reader will see that if a derivative-based method is to be used to solve (10), then a complete MDA is necessary not just at every iteration, but at every point where f or C_D or the derivatives are to be evaluated. This can be very expensive, and finite differences would be especially expensive and tricky because of loss of accuracy in computing U.

5.1.2 The Most General Formulation

For the sake of completeness, we state here the most general formulation of the MDO problem. This formulation is only for motivation and is unlikely to be useful for MDO. We state the residual and nonresidual forms together. In this kitchen sink formulation, the optimization variables are $X = (X_D, X_M, X_\mu, X_U)$ and all the conditions for a full MDA are included as auxiliary constraints:

$$\begin{array}{l} \text{minimize } \ f(X_D, X_U) \\ \text{with respect to } \ X_D, X_M, X_\mu, X_U \\ \text{subject to } \ C_D(X_D, X_U) \geq 0 \\ \text{and } \ X_M - E(X_D, X_\mu) = 0 \ \text{ and } \ X_\mu - F(X_D, X_U) = 0 \ , \\ \text{and either } \ W(X_D, X_M, X_U) = 0 \ \text{ or } \ X_U - A(X_D, X_M) = 0 \ . \end{array} \tag{11}$$

5.1.3 All-at-Once (AAO) Formulation

Now we will consider some interesting formulations between the two extremes of two preceding MDO formulations. The first we call the all-at-once (AAO) approach. In AAO, we do not seek to obtain feasibility for the analysis problem in any sense (individual discipline, multidisciplinary, or even for single equations within a discipline) until optimization convergence is reached. In a way, the optimizer does not "waste" time trying to achieve feasibility when far from an optimum. We take as explicit variables $X = (X_D, X_U)$ and write the formulation in terms of the implicit variable $M(X)$ with the interdisciplinary mapping G as its defining relation.

$$\begin{array}{ll} \text{minimize} & f(X) \quad \text{with respect to } X = (X_D, X_U) \\ \text{subject to} & C_D(X) \geq 0 \\ & C_{aux}(X) \triangleq W(X_D, M(X), X_U) = 0 \ , \end{array} \tag{12}$$

where $M(X) = G(X)$.

The drawback to (12) is that for practical problems it will generally involve a very large number of constraints (the discrete equations from all of the analysis disciplines), and an even larger number, $n_D + \sum_i n_{U_i}$, of optimization variables. Additionally, some of the constraints may not be very smooth. 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 W to attain feasibility. Generally, this means that the solution method must contain all of the special techniques that every 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 in the case where the constraints represent extremely nonlinear PDE, as in aerodynamics.

5.1.4 Individual Discipline Feasible (IDF) Formulation

Another way to avoid a complete MDA for every function value is to use an IDF formulation like (13). 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. Between these extremes lie other possibilities that amount to specific decompositions of the work between analysis codes and the optimizer. One such possibility, 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 data.

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. The rest of this section describes IDF methods.

The next formulation is the first instance of an IDF approach. The relation that defines the implicit variable $U(X)$ is just the nonresidual form of (1). Thus, each individual discipline is feasible at every optimization iteration. In this method, M is replaced by an explicit surrogate X_M and the interdisciplinary mapping becomes an auxiliary constraint. The explicit variables are $X = (X_D, X_M)$.

$$\begin{array}{ll} \text{minimize} & f(X_D, U(X)) \\ \text{with respect to} & X = (X_D, X_M) \\ \text{subject to} & C_D(X_D, U(X)) \geq 0 \\ & C_{aux}(X) \triangleq X_M - G(X_D, U(X)) = 0\ . \end{array} \tag{13}$$

where $U(X) = A(X)$. There are $n_D + \sum_i n_{M_i}$ optimization variables in this "uncompressed" IDF approach.

Notice that an evaluation of $U(X) = A(X)$ involves executing all the single discipline analysis codes with simultaneously available multidisciplinary data X. Therefore, these very expensive computations can be done independently, and communication costs are likely to be negligible in comparison. Furthermore, the analysis codes vary widely in the types of computations to be done and will generally be suitable for different hardware environments. Thus a heterogeneous network of computers may be particularly well-suited for this formulation.

The drawback to the particular IDF method (13) is the large number of optimization variables. As mentioned earlier, we can take advantage of the data compression $\mu_{ij} = F_{ij}(X_D, U_j)$ and elevate μ rather than M to be explicit variables:

$$\begin{array}{ll} \text{minimize} & f(X_D, U(X)) \\ \text{with respect to} & X = (X_D, X_\mu) \\ \text{subject to} & C_D(X_D, U(X)) \geq 0 \\ & C_{aux}(X) \triangleq X_\mu - F(X_D, U(X))) = 0. \end{array} \tag{14}$$

where $U(X) = A(X_D, E(X))$. Thus, the advantage of this "compressed" or "low-bandwidth" IDF formulation is that the optimizer controls possibly the fewest explicit variables of any IDF formulation, namely $n_D + \sum_i n_{\mu_i}$.

It is possible to write many more permutations, but we will introduce only one more: the possibility of sequencing the individual disciplines.

5.1.5 Sequenced IDF formulations

In the IDF formulations presented above, the interdisciplinary mapping (coupling) variables sent to each discipline from the other disciplines were made optimization variables and associated auxiliary constraints were imposed. We can create IDF formulations where only some of the coupling variables are optimization variables and the remainder are the actual computed analysis values. For example, consider a two discipline problem such as the aeroelastic example. The computations in the above IDF method could be sequenced such that one of the analyses is completed prior to starting the other one. Since the inputs to the second analysis would then be available, there would be no need for the optimization variables and constraints for the associated interdisciplinary mapping variables from the first to the second discipline. 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. We interpret the formulation represented by equation (12) in [22] as a sequenced IDF formulation, and so this is not unique to the present work.

5.1.6 Feasible point formulations vs. feasible point algorithms

We think it worthwhile to discuss briefly the distinction between the generalized reduced gradient (GRG) approach, which corresponds to MDF, and a gradient restoration approach applied to a full-space problem like AAO. We do this in order to prevent the misunderstanding that the formulations we have presented are not really different, but are instead just different implementations of algorithms applied to a single formulation.

Some nonlinear programming algorithms, such as gradient restoration methods, approach optimality along a feasible path by following each optimization step with a step to restore feasibility. This is very different from a generalized reduced gradient approach which maintains feasibility not just for each iterate, but for any pair X_D, U that ever appears in any context in the algorithm. In other words, the GRG approach eliminates U from the optimization calculations by using the implicitly defined function $U(X_D)$ in its stead.

If we apply a gradient restoration method to an AAO formulation, then at each optimization iteration the optimization algorithm would first take a step in the full space to obtain a complete new X. This would be followed by a so-called restoration step which would consist here of an MDA to replace the X_U part of the AAO optimization iterate X_D, X_U with $U(X_D)$. The next optimization iteration is started from the multidisciplinary feasible point $X = (X_D, U(X_D))$ satisfying (4).

However, such an approach to the AAO formulation is not the same as MDF because X_U is treated as independent of X_D for the purpose of setting the new iterate's X_D. In the MDF or GRG approach, X_D is the only variable in the optimization iteration. This means that the derivatives or sensitivities required in the MDF formulation must be computed with arguments X_D and values of all the system variables that correspond to an MDA solution for that value of X_D.

5.2 Formulations specialized to the aeroelastic MDO problem

To further elucidate the formulation ideas, we show how the general formulations apply to the specific case of aeroelastic MDO.

First is the standard MDF formulation (10)

$$\begin{array}{ll} \text{minimize} & f(X_D, U_A(X_D), U_S(X_D)) \quad \text{with respect to } X_D \\ \text{subject to} & C_D(X_D, U_A(X_D), U_S(X_D)) \geq 0, \\ \text{where} & U_A(X_D) = A_A(X_D, G_{AS}(X_D, U_S(X_D))) \\ & U_S(X_D) = A_S(X_D, G_{SA}(X_D, U_A(X_D))) \ . \end{array} \tag{15}$$

Figure 3 illustrates the MDF formulation. Notice that while Figure 3 provides some detail about the computations of U_S and U_A, the aeroelastic analysis is a "black box" from the perspective of the optimization code.

If analysis residuals are available, then one might try to avoid so many costly MDA computations by an All-At Once, or AAO, formulation with U_A and U_S made explicit. Figure 4 illustrates the the AAO formulation. The AAO optimization problem is

$$\begin{array}{ll} \text{minimize} & f(X) \\ \text{with respect to} & X = (X_D, X_{U_A}, X_{U_S}) \\ \text{subject to} & C_D(X) \geq 0 \\ & W_A(X_D, M_{AS}(X), X_{U_A}) = 0 \\ & W_S(X_D, M_{SA}(X), X_{U_S}) = 0 \ , \end{array} \tag{16}$$

where $M_{AS}(X) = G_{AS}(X_D, X_{U_S})$ and $M_{SA}(X) = G_{SA}(X_D, X_{U_A})$.

Other "all-at-once" (AAO) formulations for design optimization problems have been mentioned in the literature for aerodynamic optimization (e.g., [5, 9, 13, 21]), structural optimization (e.g., [6]), chemical process control, and control and inverse problems (e.g., [16, 20]). In [13] this approach is called the "one-shot" method, and in [6] it is called "simultaneous analysis and design." In [5] the authors discuss how AAO can be remarkably efficient for aerodynamic optimization, provided some computational difficulties can be overcome.

The rest of this section describes two aeroelastic IDF methods. We reiterate the essence of IDF: 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 correspond to the loads sent to the structures and the displacements predicted by the structural analysis correspond to the geometry sent to the aerodynamics code. Again, we remind the reader that one could follow each optimization step by performing a feasibility restoring MDA, but the optimization problem being solved would still be an IDF and not an MDF formulation.

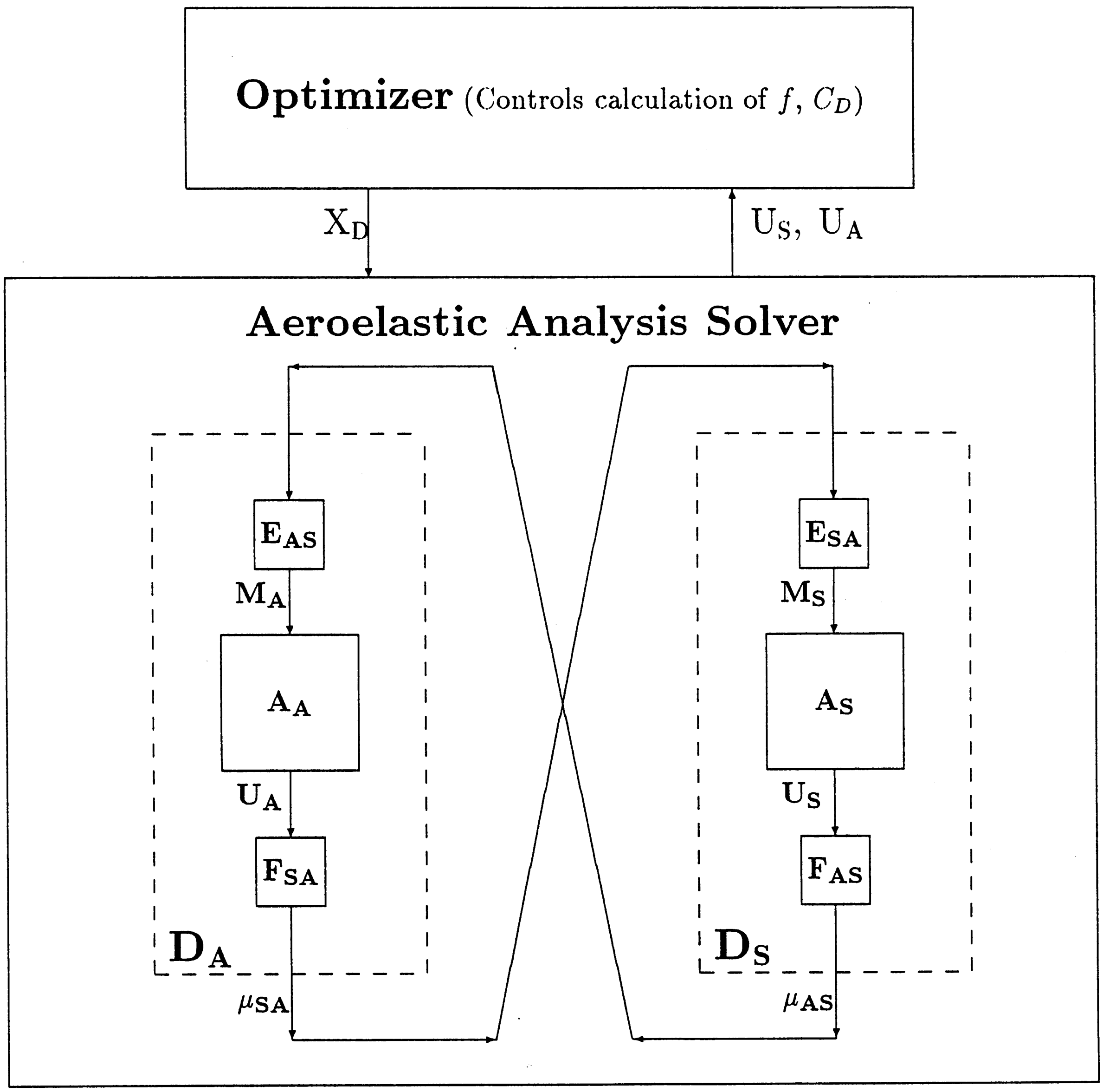

FIGURE 3: Multidisciplinary Feasible (MDF) Method

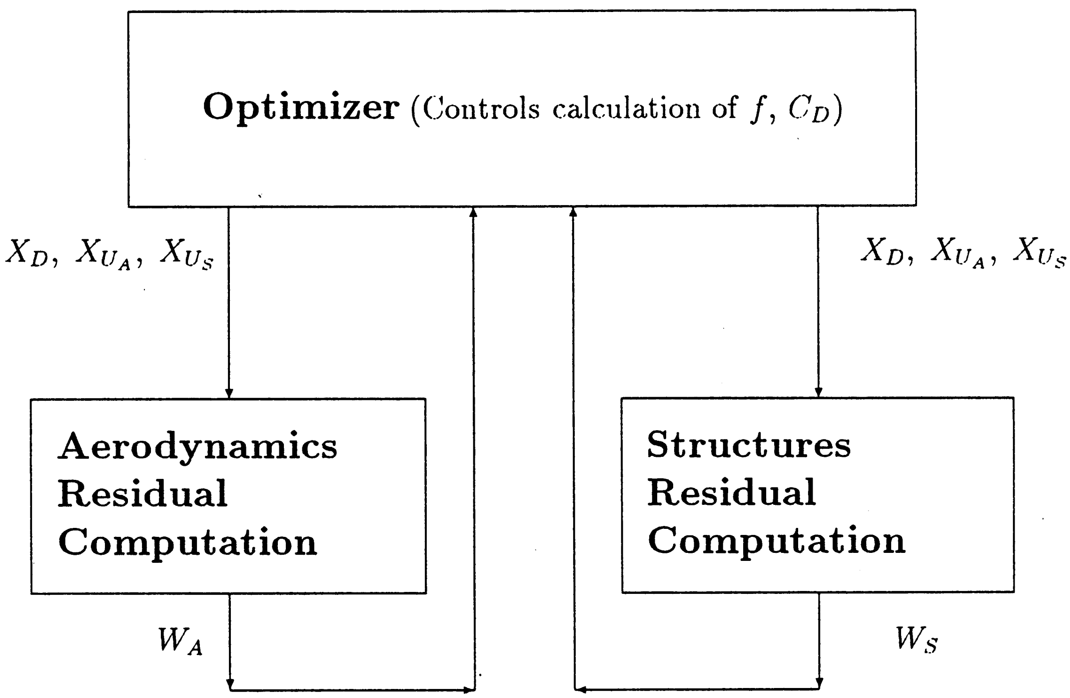

FIGURE 4: All-at-Once (AAO) Method

The "uncompressed" IDF formulation is

$$\begin{array}{ll} \text{minimize} & f(X_D, U_A(X), U_S(X)) \\ \text{with respect to} & X = (X_D, X_{M_{AS}}, X_{M_{SA}}) \\ \text{subject to} & C_D(X_D, U_A(X), U_S(X)) \geq 0 \\ & C_{AS} \equiv X_{M_{AS}} - G_{AS}(X_D, U_S(X)) = 0 \\ & C_{SA} \equiv X_{M_{SA}} - G_{SA}(X_D, U_A(X)) = 0\ . \end{array} \tag{17}$$

where $U_A(X) = A_A(X_D, X_{M_{AS}})$ and $U_S(X) = A_S(X_D, X_{M_{SA}})$.

The low-bandwidth IDF formulation is

$$\begin{array}{ll} \text{minimize} & f(X_D, U_A(X), U_S(X)) \\ \text{with respect to} & X = (X_D, X_{\mu_{AS}}, X_{\mu_{SA}}) \\ \text{subject to} & C_D(X_D, U_A(X), U_S(X)) \geq 0 \\ & C_{AS} \equiv X_{\mu_{AS}} - F_{AS}(X_D, U_S(X)) = 0 \\ & C_{SA} \equiv X_{\mu_{SA}} - F_{SA}(X_D, U_A(X)) = 0. \end{array} \tag{18}$$

where $U_A(X) = A_A(X_D, E_{AS}(X_D, X_{\mu_{AS}}))$ and $U_S(X) = A_S(X_D, E_{SA}(X_D, X_{\mu_{SA}}))$. Figure 5 shows the flow of information for this low-bandwidth IDF formulation.

6 Derivative Requirements for MDO

We anticipate that most MDO efforts will involve derivative-based optimization algorithms. For this reason we now will discuss the derivatives required in the MDF, IDF, and AAO

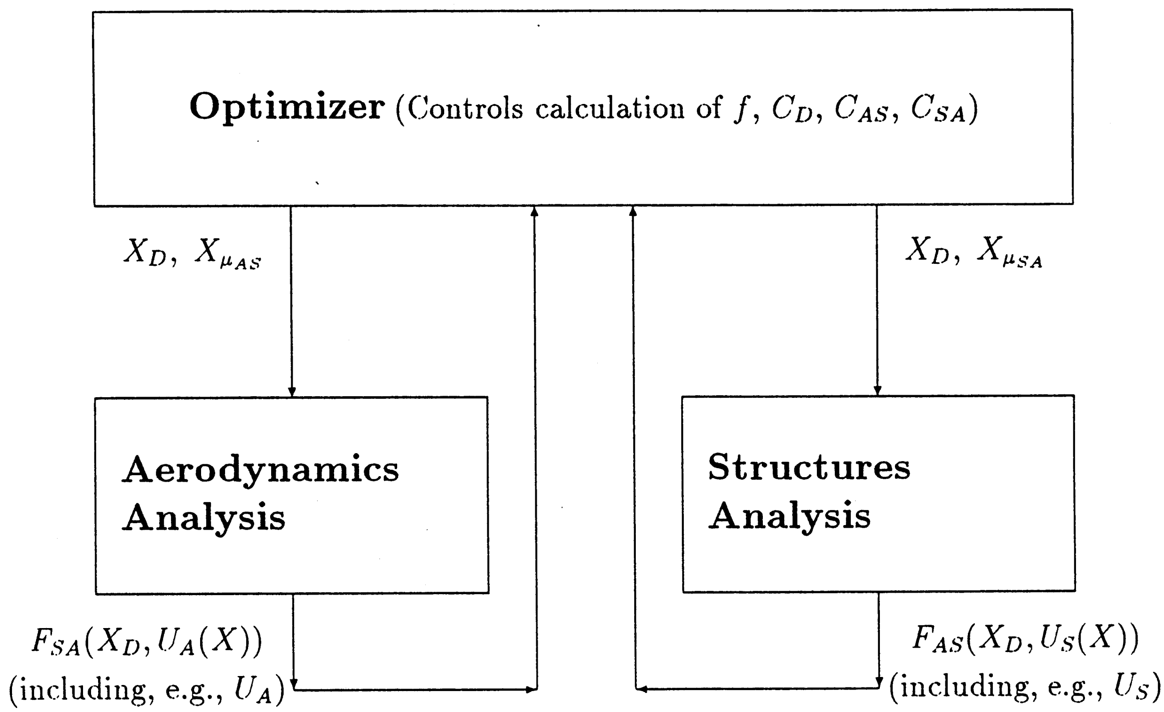

FIGURE 5: Individual discipline feasible method

formulations that we have presented. If one looks in the previous section, all the formulations given either use the analysis residuals as a constraint, or else the analysis code solution mapping is used to define U as an implicit variable. Thus, any derivative-based algorithm will require either the derivative of the analysis residuals or of the solution operator.

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 and IDF formulations have the advantage that they use the specialized software A_i that has been developed for solving the individual discipline equations. But there is a price to be paid for using the existing software; the MDF and IDF formulations must differentiate the solution operators implemented by the single discipline solvers.

The most daunting task is to obtain these solution sensitivities. Most attempts at MDO have used finite-difference approximations to derivatives. This certainly finesses the issue, but because of problems with accuracy and expense, we believe that a practical alternative to finite differences must be found if MDO is to become an everyday engineering tool. There seem to be two alternatives: analytic approaches (implicit differentiation, sensitivity equations, adjoint equation solution) and automatic differentiation.

Even though there is considerable research interest in analytic methods for sensitivity or gradient calculations [2, 1, 8, 6, 7, 10, 11], few analysis codes in engineering use today provide the required derivatives. We hope that automatic differentiation will provide an alternative approach to retrofit existing codes to produce the derivatives.

Automatic differentiation should not be confused with symbolic differentiation. In ADIFOR [17], the automatic differentiation tool with which we are the most familiar, the definition of the function is given as a standard Fortran program. The output from ADIFOR is a program that duplicates the computation of the original program, and in addition, it includes code to compute the sensitivities of indicated outputs with respect to indicated outputs. The sensitivities are computed with the same accuracy as the quantities whose partial derivatives they represent.

We finish our discussion of the problems of finding derivatives with a brief discussion of the derivatives needed by the various formulations. There seems little point to laboring through the general case, and so we will restrict ourselves to the aeroelastic example problem. In the subsections that follow we give the derivatives of the constraints. However, the objective function f and the design constraints C_D depend on the same parameters. Thus, the gradient of the objective function is just the transpose of the block row of the constraint Jacobian corresponding to C_D, with C_D replaced by f.

6.1 Derivatives required for the MDF formulation

For MDF optimization, Sobieski [12] gives a complete presentation of the alternative approaches, but our MDO model includes the fit and evaluate routines and so the form is slightly different here.

Using the MDF formulation given by (15), the linearized constraint residual is, noting $X = [X_D]$,

$$\left[\ C_D^{(c)}\ \right] + \left[\ \frac{\partial C_D}{\partial X_D} + \frac{\partial C_D}{\partial U_A}\frac{\partial U_A}{\partial X_D} + \frac{\partial C_D}{\partial U_S}\frac{\partial U_S}{\partial X_D}\ \right]\left[\ \Delta X_D\ \right], \tag{19}$$

where $C_D^{(c)}$ is the value of the constraints at the current approximation $X_D^{(c)}$ to the solution of the MDO problem. Remember, for the MDF formulation, $X_D^{(c)}$ is a full MDA solution point. The coefficient matrix in (19) represents the Jacobian of C_D with respect to the design variables X_D.

Computing the partial derivatives $\partial f/\partial X_D$, $\partial f/\partial U_\alpha$, $\partial C_D/\partial X_D$, and $\partial C_D/\partial U_\alpha$ for $\alpha = A, S$ is generally easy; computing the solution sensitivities $\partial U_\alpha/\partial X_D$ is generally hard. One way to obtain these sensitivities is to form and solve a linear system as follows. Notice that since we need the derivatives at a point $X_D^{(c)}$ for which (7) holds, we can apply implicit differentiation to the appropriate residual equations to obtain:

$$\begin{aligned}\frac{\partial W_A(X_D,M_{AS},U_A)}{\partial X_D} + \frac{\partial W_A(X_D,M_{AS},U_A)}{\partial M_{AS}}&\left[\frac{\partial G_{AS}}{\partial X_D} + \frac{\partial G_{AS}}{\partial U_S}\frac{\partial U_S}{\partial X_D}\right]\\ &+\frac{\partial W_A(X_D,M_{AS},U_A)}{\partial U_A}\frac{\partial U_A}{\partial X_D} = 0\end{aligned}$$

and

$$\begin{aligned}\frac{\partial W_S(X_D,M_{SA},U_S)}{\partial X_D} + \frac{\partial W_S(X_D,M_{SA},U_S)}{\partial M_{SA}}&\left[\frac{\partial G_{SA}}{\partial X_D} + \frac{\partial G_{SA}}{\partial U_A}\frac{\partial U_A}{\partial X_D}\right]\\ &+\frac{\partial W_S(X_D,M_{SA},U_S)}{\partial U_S}\frac{\partial U_S}{\partial X_D} = 0\end{aligned} \tag{20}$$

where we have used the fact that $M = G(X_D, U)$. Every derivative in (20) is assumed to be available except the derivatives of U. Thus, we can gather terms to obtain a linear system that can be solved to obtain the sought for U derivatives:

$$\begin{bmatrix} \frac{\partial W_A}{\partial U_A} & \frac{\partial W_A}{\partial M_{AS}}\frac{\partial G_{AS}}{\partial U_S} \\ \frac{\partial W_S}{\partial M_{SA}}\frac{\partial G_{SA}}{\partial U_A} & \frac{\partial W_S}{\partial U_S} \end{bmatrix}\begin{bmatrix} \frac{\partial U_A}{\partial X_D} \\ \frac{\partial U_S}{\partial X_D} \end{bmatrix} = -\begin{bmatrix} \frac{\partial W_A}{\partial X_D} + \frac{\partial W_A}{\partial M_{AS}}\frac{\partial G_{AS}}{\partial X_D} \\ \frac{\partial W_S}{\partial X_D} + \frac{\partial W_S}{\partial M_{SA}}\frac{\partial G_{SA}}{\partial X_D} \end{bmatrix}. \tag{21}$$

If the residuals are not available, then in a similar way, we can differentiate (7) and rearrange terms to obtain:

$$\begin{bmatrix} I & -\frac{\partial U_A}{\partial M_{AS}}\frac{\partial G_{AS}}{\partial U_S} \\ -\frac{\partial U_S}{\partial M_{SA}}\frac{\partial G_{SA}}{\partial U_A} & I \end{bmatrix} \begin{bmatrix} \frac{\partial U_A}{\partial X_D} \\ \frac{\partial U_S}{\partial X_D} \end{bmatrix} = \begin{bmatrix} \frac{\partial U_A}{\partial X_D} + \frac{\partial U_A}{\partial M_{AS}}\frac{\partial G_{AS}}{\partial X_D} \\ \frac{\partial U_S}{\partial X_D} + \frac{\partial U_S}{\partial M_{SA}}\frac{\partial G_{SA}}{\partial X_D} \end{bmatrix}, \tag{22}$$

where we have used (7) to replace partial derivatives of A_A and A_S by partial derivatives of U_A and U_S, respectively. It is important to remember here that *all* of these partials must be evaluated at $X_D^{(c)}$, a multidisciplinary feasible point.

6.2 Derivatives required for the AAO formulation

The AAO formulation has much easier derivative requirements because the derivatives do not have to be evaluated at MDA solutions. In fact, for formulation (12), there is no feasibility required at arbitrary evaluation points. The linearized constraint residual is, noting $X = [X_D, X_U]$ and $X_U = [X_{U_A}, X_{U_S}]$,

$$\begin{bmatrix} C_D^{(c)} \\ W_A^{(c)} \\ W_S^{(c)} \end{bmatrix} + \begin{bmatrix} \frac{\partial C_D}{\partial X_D} & \frac{\partial C_D}{\partial X_{U_A}} & \frac{\partial C_D}{\partial X_{U_S}} \\ \frac{\partial W_A}{\partial X_D} & \frac{\partial W_A}{\partial X_{U_A}} & \frac{\partial W_A}{\partial X_{U_S}} \\ \frac{\partial W_S}{\partial X_D} & \frac{\partial W_S}{\partial X_{U_A}} & \frac{\partial W_S}{\partial X_{U_S}} \end{bmatrix} \begin{bmatrix} \Delta X_D \\ \Delta X_{U_A} \\ \Delta X_{U_S} \end{bmatrix}, \tag{23}$$

where $(C_D^{(c)}, W_A^{(c)}, W_S^{(c)})^T$ is the residual of the constraints at the current point.

The blocks $\partial W_A/\partial X_{U_A}$ and $\partial W_S/\partial X_{U_S}$ in (23) are the Jacobians that would appear in Newton solvers for the two disciplines, respectively. The derivatives $\partial W_A/\partial X_D$, $\partial W_A/\partial X_{U_S}$, $\partial W_S/\partial X_D$, and $\partial W_S/\partial X_{U_A}$ represent the sensitivities of the analysis discipline equation residuals to their inputs from other disciplines. We assume that these derivatives are available.

6.3 Derivatives required for the IDF formulations

The linearized constraint residual for the IDF formulation (17) is

$$\begin{bmatrix} C_D^{(c)} \\ C_{AS}^{(c)} \\ C_{SA}^{(c)} \end{bmatrix} + J \begin{bmatrix} \Delta X_D \\ \Delta X_{M_{AS}} \\ \Delta X_{M_{SA}} \end{bmatrix}. \tag{24}$$

where

$$J = \begin{bmatrix} \frac{dC_D}{dX_D} & \frac{\partial C_D}{\partial U_A}\frac{\partial U_A}{\partial X_{M_{AS}}} & \frac{\partial C_D}{\partial U_S}\frac{\partial U_S}{\partial X_{M_{SA}}} \\ -\frac{\partial G_{AS}}{\partial X_D} - \frac{\partial G_{AS}}{\partial U_S}\frac{\partial U_S}{\partial X_D} & I & -\frac{\partial G_{AS}}{\partial U_S}\frac{\partial U_S}{\partial X_{M_{SA}}} \\ -\frac{\partial G_{SA}}{\partial X_D} - \frac{\partial G_{SA}}{\partial U_A}\frac{\partial U_A}{\partial X_D} & -\frac{\partial G_{SA}}{\partial U_A}\frac{\partial U_A}{\partial X_{M_{AS}}} & I \end{bmatrix}, \tag{25}$$

$$\frac{dC_D}{dX_D} = \frac{\partial C_D}{\partial X_D} + \frac{\partial C_D}{\partial U_A}\frac{\partial U_A}{\partial X_D} + \frac{\partial C_D}{\partial U_S}\frac{\partial U_S}{\partial X_D}, \tag{26}$$

and $\left(C_D^{(c)}, C_{AS}^{(c)}, C_{SA}^{(c)}\right)^T$ is the value of the constraints at the current point. Derivatives similar to the above can be derived for the low-bandwidth IDF formulation (18).

The expensive derivatives for the IDF method are those of the form $\partial U_i/\partial X_D$, $\partial U_i/\partial X_{M_{ij}}$, and $\partial U_i/\partial X_{\mu_{ij}}$, which are all sensitivities of the individual discipline analysis solutions with respect to either uncompressed or compressed analysis inputs. Note that the derivatives required for the IDF formulation are the same as those required in (22) and by Sobieski [12] (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.*

7 Considerations in Choosing a Formulation

In this section, we will discuss briefly some important issues related to choosing an MDO formulation for a specific problem: multipoint design, and the opportunities to exploit parallel computing.

Because MDO problems come in all sizes and shapes, and because there are very often discrete optimization variables, it is impossible to say much about how to choose an optimization algorithm for a general MDO problem. Given a problem and an NLP code, we do offer in the next section some advice on choosing a formulation.

Generally, in MDO as elsewhere we will choose between different optimization techniques based on problem size, smoothness, derivative availability, and sparsity. Frank et al. [4] investigated the applicability of derivative-based methods (nonlinear programming techniques), response surfaces, expert systems, genetic algorithms, simulated annealing, and neural networks to MDO problems.

As one would suspect, they concluded that for problems where derivative-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 differentiable problems so that derivative-based methods can be applied. However, they recognized that this cannot always be done.

They also discuss global optimization for MDO problems. Even in this case, derivative-based methods were recommended to solve the local optimization subproblems.

7.1 Parallelism

As previously mentioned, the IDF formulation is well-suited to implementation in a heterogeneous computing environment comprised of computers suitable for each particular disciplinary analysis. Such a computing environment would enable us to take advantage of the effort made in many disciplines to optimize the performance of analysis codes on particular machines. Moreover, the computational expense we can expect inside many of the individual disciplinary analyses makes such loosely coupled parallelism natural. At the same time, this environment is possibly the only practical one for many MDO problems, since no single computer may be large enough to run the entire large MDO problem.

There is also an attractive logistical reason for exploiting such parallelism. An MDO design group made up of single discipline design groups already conducting single discipline designs could implement the IDF method in parallel on a network of the machines that already run the individual disciplines' analysis codes.

The MDF and IDF methods, since they leave intact the disciplinary analysis codes, do not lose any of the parallelism that might have been developed there by disciplinary specialists. In contrast, the AAO method is forced to provide its own approach to solving the analysis equations. Thus, an AAO implementation is on its own to provide ways to exploit parallelism. Since the algorithm to be used will probably have a huge sparse quadratic program as its computational kernel, this is quite problematic.

A similar issue arises for the IDF approach. Figure 5 shows the interdisciplinary connectivity for our model problem of aeroelastic optimization. These connections reveal, to a degree, the patterns of communication in a parallel implementation of the optimization solution. For instance, this diagram shows the independence of the bulk of the computation needed to compute function and constraint values. However, there is an important level of detail that is omitted from this diagram. We have not indicated the patterns of communication inside the computational kernel of the optimizer. Depending on the size of the problem and the nature of the optimization algorithm, this may be significant to an efficient parallel implementation.

If the optimization algorithm has as its primary computational subtask the solution of a relatively small model problem (one in which the model gradient and Hessian can be assembled), then we likely need not worry about the patterns of communication inside the optimizer. On the other hand, for very large MDO problems, the pattern of computation and communication inside the optimization algorithm should reflect that of the problem as it is presented in Figure 5. Otherwise, the computations inside the optimization and the parallelism of the coupled disciplines may be at odds, limiting the boon we can expect from parallelism in the IDF approach.

In particular, this means that the optimization block in Figure 5 should not be viewed as a monolithic block residing on a single computational unit. Instead, it may be distributed among the computational units together with the individual disciplines. This would enable the optimization algorithm to take advantage of the block structure of the MDO problem in the execution of the optimization subtasks, while respecting the distribution and flow of information inherent to the MDO problem.

7.2 Multipoint design

MDO problems will often involve design over several cases, or design points. In the aeroelastic problem, for example, there may be stress constraints for several flight conditions such as pull-up or dive maneuvers. There may also be minimum performance requirements for off-design values of velocity, altitude, etc. In fact, different analysis codes may be used for different design points. For example, a low-fidelity aerodynamics analysis code may be acceptable for computing pressures for a dive maneuver, while a high-fidelity aerodynamics analysis code may be required for computing drag and lift at cruise.

MDO over multiple design points can be readily couched in the formulations presented here by considering the analyses at each point to be a separate "discipline".

8 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. These hypotheses are supported by the experimental results shown in [23, 18].

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

	All-at-once (AAO)	(Compressed) Individual Discipline Feasible (IDF)	Multidisciplinary Feasible (MDF)
Use of existing analysis 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	Multidisciplinary feasibility 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 sensitivities required.)	$n_D + \sum_i n_{U_i}$	$n_D + \sum_i n_{\mu_i}$	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 Feasible (IDF)	Multidisciplinary 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	Unknown	High	Medium

TABLE 2: Comparison of predicted performance

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 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 conclusively 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 then to correct via iterative refinement. For example, this approach has been used in multidisciplinary design of helicopter rotors [14]. We observe that this approach dovetails well with the IDF approach, where an existing multidisciplinary analysis procedure can be viewed as "one discipline," and information of higher fidelity for a single analysis code can be the "second discipline" [19]. The iterative refinement outer loop could build a response surface model of the higher fidelity code. The iterative refinement optimization loop would then use the IDF method with the multidisciplinary analysis as one discipline and the response surface model as the second discipline.

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.

References

[1] Baysal, O. and E.M. Eleshaky. Aerodynamic design using sensitivity analysis and computational fluid dynamics. Technical Report AIAA-91-0471, AIAA, January, 1991. Presented at the 29th Aerospace Sciences Meeting, January 7-10, 1991, Reno, Nevada.

[2] Borggaard,J., J.A. Burns, E.M. Cliff, and M. Gunzburger. Sensitivity calculations for 2d, inviscid, supersonic forebody problem. Technical Report 93-13, ICASE, March, 1993.

[3] Cramer, E.J., P.D. Frank, G.R. Shubin, J.E. Dennis, Jr., and R.M. Lewis. On alternative problem formulations for multidisciplinary design optimization. AIAA-92-4752, 4th Symposium on Multidisciplinary Analysis and Optimization, September 21-23, 1992, Cleveland, OH.

[4] 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.

[5] 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.

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

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

[8] Korivi, V., A. Taylor, III, P. Newman and G. Hou, and H. Jones. An approximately factored incremental strategy for calculating consistent discrete CFD sensitivity derivatives. AIAA-92-4746, 4th Symposium on Multidisciplinary Analysis and Optimization, September 21-23, 1992, Cleveland, OH.

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

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

[11] 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.

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

[13] 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.

[14] Young, D.K. and F.J. Tarzanian Jr. Structural optimization and Mach scale test validation of a low vibration rotor. 47th Annual Forum of the American Helicopter Society, Phoenix, Arizona, 1991.

[15] AIAA Technical Committee on Multidisciplinary Design Optimization (MDO). White paper on current state of the art. American Institute of Aeronautics and Astronautics, 1991.

[16] H.T. Banks and K. Kunisch. *Estimation techniques for distributed parameter systems.* Birkhauser, 1989.

[17] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland. ADIFOR: generating derivative codes from Fortran programs. Technical Report CRPC-TR91185, Center for Research on Parallel Computation, December, 1991.

[18] J.E. Dennis, Guangye Li, and Karen Williamson. Optimization algorithms for parameter identification. Technical Report CRPC-TR92277, Center for Research on Parallel Computation, January, 1992.

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

[20] F.-S. Kupfer and E. W. Sachs. A prospective look at SQP methods for semilinear parabolic control problems. In *Optimal control of partial differential equations: proceedings of the IFIP WG 7.2 International Conference*, pages 145–157, 1991.

[21] C.E. Orozco and O. N. Ghattas. Massively parallel aerodynamic shape optimization. preprint.

[22] R. T. Haftka, J. Sobieszczanski–Sobieski, and S. L. Padula. On options for interdisciplinary analysis and design optimization. *Structural Optimization*, 4(2):65–74, 1992.

[23] Shubin, G.R. Application of alternative multidisciplinary optimization formulations to a new model problem for static aeroelasticity. Technical Report BCSTECH-93-022, Boeing Computer Services, August, 1993.