

MULTI-DISCIPLINARY DESIGN OPTIMIZATION ARCHITECTURES

1. Introduction

Multi-disciplinary Design Optimization (MDO) is an emerging design enabler, which addresses the issue of carrying out formal design optimization of complex, coupled systems like aerospace vehicles. A system can be thought of comprising of several “disciplines” or “components” or “modules”. A coupled system occurs either when a design variable is shared by more than one discipline or when the output of one discipline is needed as an input by another discipline. A schematic of a three discipline coupled system is shown in figure 1.1 and the notations used are explained in the next chapter. If the system is an aircraft, then aerodynamics, structures and performance could be the three disciplines. The modules comprising the system could also result in coupling. For example, the fore-body design of a Hypersonic vehicle strongly influences the design of the intake module. In the design of such a complex, coupled engineering system, there is an increasing demand to include the effects of interactions among the various disciplines right at the conceptual and preliminary design stages. The idea is to study the combined effect of the disciplines on a system level performance merit. In such a scenario, it is very much likely that the individual disciplines may be operating at sub-optimal conditions, but the synergistic effect of all the disciplines ensures that the system level objective is optimized. The greatest potential to beneficially use the trade-off among the various disciplines occurs at the preliminary design phase, as the system is still open to improvement in definition. Once the design is in advanced stage, it may prove costly to introduce any modifications in order to improve the design. However, to realize this potential, it is important to ensure that the analysis models, which support the design, are not simplistic. Otherwise, the preliminary results may contradict the results obtained at the detailed phase. The emerging field of MDO looks into the issue of efficiently implementing a coupled design process incorporating appropriate analysis fidelity levels. One can formally define MDO to mean the “*systematic approach to optimization of complex, coupled engineering systems*” wherein multidisciplinary refers to different aspects/disciplines that must be included in the system design activity[1]. It is evident from the definition that there are more than one competing disciplines involved in the design of the system. MDO allows disciplinary groups to analyze and design in parallel, with a certain degree of autonomy. However, since the disciplines are coupled, a system level coordination is

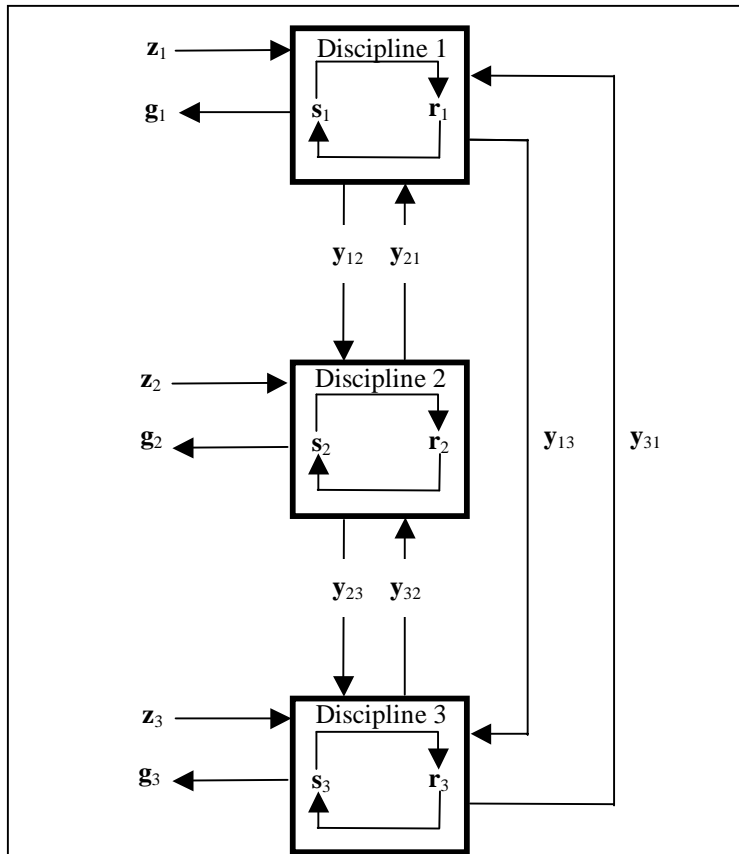


Fig.1.1 Schematic of Three Discipline Coupled

required. The term **MDO Architecture** encompasses the activity of posing the design problem as a set of mathematical statements amenable to solution. This can be referred to as “formulation” of the design problem. The efficient integration of the system analysis and an automated decision making process i.e., optimization is the eventual goal in the development of a MDO architecture. A natural way of posing the problem is to complete the coupled system analysis and return the solution vector to the optimizer for each

iteration of optimization. This is referred to as the Multi-Disciplinary Feasible (MDF) formulation. The term “feasible” here means that all the disciplines have been solved and the solution satisfies the interdisciplinary couplings at all iterations of the optimization. In such a strategy, the optimization process is very expensive since a complete coupled system analysis is being carried out at all iterations. This becomes a practical drawback especially when high fidelity tools like CFD are used for analysis. The expense of implementing this straightforward optimization approach has motivated research into alternative formulations. Another reason for seeking alternative formulations is to incorporate features of organizational structure in the formulation. Typically, a design organization is decomposed along disciplinary boundaries like Aerodynamics, Propulsion, Structures etc. Independent discipline activities are coordinated through Project Offices. Several MDO architectures or formulations have emerged over the last decade. Each formulation differs from the other in the manner in which the interdisciplinary couplings are handled, while at the same time trying to achieve individual discipline autonomy. Thus it is possible to cast a given design problem in several formulations. It is important to note, however, that the solutions obtained from alternative formulations must also be solutions in the Multi-Disciplinary Feasible formulation.

This report provides a description of the various MDO Architectures and a comparison is drawn among them . The mathematical statements of the various architectures are also given . It may be noted that different notations have been used to describe the same type of variable by various groups of researchers in the field of MDO. This report attempts to present all the architectures using a unified notation. The report is organized as follows:

The terminology associated with MDO architecture/formulations is introduced Chapter 2. The evolution of MDO formulations is discussed in Chapter 3. The basis for classification of MDO formulations and the taxonomy of formulations are introduced in Chapter 4. Chapter 5 and 6 present discussions on Single and Bi-level formulations. A comparative assessment of computational performance of two formulations is presented in Chapter 7. Finally, the conclusions are stated in Chapter 8.

Chapter 2: Terminology

In this chapter, the terminology associated with MDO architectures [2,3,4,5] is explained.

A general non-linear programming problem is stated as follows:

$$\begin{aligned} &\text{Minimize } f(\mathbf{Z}) \\ &\text{Subject to } \mathbf{g}(\mathbf{Z}) \leq 0 ; \mathbf{h}(\mathbf{Z}) = 0 \end{aligned} \quad (2.1)$$

An engineering optimization problem assumes the following form

$$\begin{aligned} &\text{Minimize } f(\mathbf{Z}, \mathbf{S}(\mathbf{Z})) \\ &\text{Subject to } \mathbf{g}(\mathbf{Z}, \mathbf{S}(\mathbf{Z})) \leq 0 ; \mathbf{h}(\mathbf{Z}, \mathbf{S}(\mathbf{Z})) = 0 \end{aligned} \quad (2.2)$$

where $\mathbf{S}(\mathbf{Z})$ is a state variable vector and is obtained through an analysis of the form

$$\mathbf{A}(\mathbf{Z}, \mathbf{S}(\mathbf{Z})) = 0 \quad (2.3)$$

Analysis may be viewed as satisfying conservation laws for the state variables for the given \mathbf{Z} . This often has a non-linear form and it is solved iteratively until convergence. The notations used above are explained below.

Design Variables

Variables, which describe each design, and distinguish it from others, are called design variables. For example, wing-span, chord, thickness/chord can be a few design variables in the design of an aerospace vehicle. Some of design variables are shared by more than one discipline, while others are local to one discipline. The vector of the design variables will be denoted as \mathbf{Z} and its elements are as follows:

$$\mathbf{Z} = \mathbf{Z}_S \cup \mathbf{Z}_L$$

$$\mathbf{Z}_S = \cup \mathbf{z}_{si}$$

$$\mathbf{Z}_L = \cup \mathbf{z}_{Li}$$

where, the shared variables will be denoted as \mathbf{Z}_S and the local variables as \mathbf{Z}_L . Only a part of the vector \mathbf{Z}_S may be needed by discipline 'i'. The shared variables required in discipline 'i' will be denoted as \mathbf{z}_{si} . Hence, we can write $\mathbf{Z}_S = \cup \mathbf{z}_{si}$. However, the shared variable vectors need not be mutually disjoint i.e., $\mathbf{z}_{si} \cap \mathbf{z}_{sj} \neq 0$, for $i \neq j$. \mathbf{Z}_L is the union of all the local variables for each discipline i.e., $\mathbf{Z}_L = \cup \mathbf{z}_{Li}$, where \mathbf{z}_{Li} denotes the variables local to discipline 'i'. Note that $\mathbf{z}_{Li} \cap \mathbf{z}_{Lj} = 0$, for $i \neq j$. Let all the variables, including shared and local, belonging to discipline 'i' be denoted as \mathbf{z}_i . Then, $\mathbf{z}_i = \mathbf{z}_{si} + \mathbf{z}_{Li}$.

State Equations and State Variables

State equations are used to represent conservation laws of the system. $(AIC)^* \Gamma - \alpha = 0$, is an example of a State equation representing equilibrium between vortex strength and angle of attack. (AIC is Aerodynamic Influence Coefficient Matrix). Γ is an example of state variable in the above equation. State variables pertaining to discipline 'i' will be denoted as s_i and $S = \cup s_i$

Coupling functions and Coupling Variables

A function that is evaluated in discipline 'j' may be needed as an input in discipline 'k'. Such functions are termed as coupling functions and are denoted as y_{jk} . These depend on z_{Lj} , z_{sj} and s_j . Coupling variables are variables introduced in order to make the analysis of discipline 'k' independent of discipline 'j'. Thus with each coupling function, one can associate a coupling variable denoted as y^*_{jk} .

Analysis, Evaluator and Residuals

In engineering design problems, the state equations are generally, non-linear and its solution is an iterative process. Obtaining a convergent solution for this iterative process is called Analysis. On the other hand, given the design variable x and state variables s , Evaluators estimate the error in the state equations. Analyzer and Evaluator are shown schematically in figure.2.1. For example in VLM code, the vortex strength vector Γ is calculated as $\Gamma = (AIC)^{-1}\alpha$ by inverting the Aerodynamic Influence Coefficient (AIC) matrix. This is called Analysis. Instead, for a given vector α and an assumed vector Γ , the evaluator outputs the residual, given as $r = (AIC)^* \Gamma - \alpha$. Evaluators are used in many

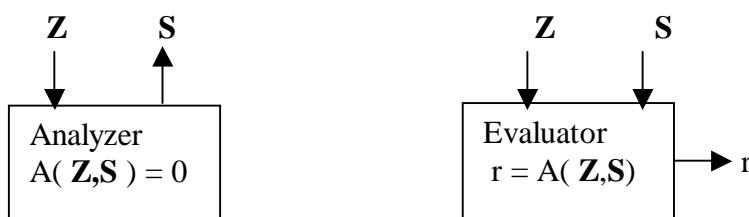


Fig.2.1 Schematic of Analysis and Evaluator

optimization strategies to reduce the computational effort.

Design Constraint and Objective function

Design constraints are prescribed by the user, to ensure that the design meets certain basic requirements. For example, range of the vehicle to be a certain minimum value. They are also formulated or prescribed to avoid failures and unacceptable behavior. They are denoted by 'g' for inequality type constraints and 'h' for equality type constraints. The objective function is generally denoted as 'f'.

3. Evolution of MDO Architectures

The evolution of MDO architecture and certain features of alternative formulations are discussed in this chapter.

In the context of Multidisciplinary Design Optimization (MDO), a complex, coupled engineering system is represented by equation (2.3) which is repeated below:

$$\mathbf{A}(\mathbf{Z}, \mathbf{S}(\mathbf{Z})) = 0 \quad (2.3)$$

In the expanded form, the above equation can be written as

$$\begin{aligned} \mathbf{A}_1(\mathbf{z}_1, \mathbf{s}_1, \mathbf{y}_{21}, \mathbf{y}_{31}, \dots, \mathbf{y}_{n1}) &= 0 \\ \mathbf{A}_2(\mathbf{z}_2, \mathbf{s}_2, \mathbf{y}_{12}, \mathbf{y}_{32}, \dots, \mathbf{y}_{n2}) &= 0 \\ \dots & \\ \dots & \\ \dots & \\ \mathbf{A}_n(\mathbf{z}_n, \mathbf{s}_n, \mathbf{y}_{1n}, \mathbf{y}_{2n}, \dots, \mathbf{y}_{n-1, n}) &= 0 \end{aligned} \quad (3.1)$$

\mathbf{A}_i in the set of equations (3.1) is coupled to all other \mathbf{A}_j ($j \neq i$), through \mathbf{y}_{ji} ($j \neq i$). If all \mathbf{y}_{ji} ($j \neq i$) are considered as inputs to \mathbf{A}_i , then \mathbf{A}_i may be treated as a set of equations in the unknown states \mathbf{s}_i . It may also be noted that the discipline design vector ' \mathbf{z}_i ' contains some variables \mathbf{z}_{si} , which are needed as input by more than one discipline and some variables \mathbf{z}_{Li} which are purely local to the discipline. Multi-disciplinary Analysis (MDA) refers to solution of the system of equations (3.1). Typically for engineering systems, solving (3.1) is very time intensive and iterative in nature. Different strategies can be adopted for solving the optimization problem (2.2) with (3.1), resulting in different MDO Architectures. Consider the

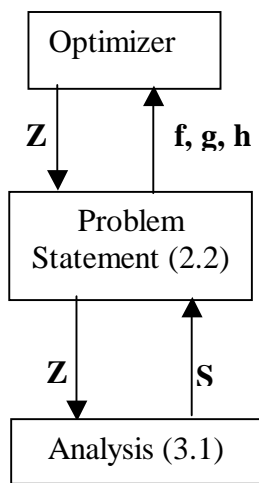


Fig.3.1 Schematic of an Engineering Design Problem using Analysis

case where (3.1) is solved to convergence. This is referred to as analysis 'analysis' [2,5]. A schematic of such a strategy is shown in figure3.1. However, in the context of optimization, it may be computationally expensive to carry out a converged analysis at every optimization iteration, when one is far away from the optimal point. Evaluators, provide an alternative way to reduce the computational expense involved in the Analysis. In this case the analysis is 'open', since the MDA equations (3.1) are not identically

satisfied but return residues \mathbf{r}_i . The state variables s_i are augmented to the design vector \mathbf{Z} . Additional equality constraints involving the residues are augmented to the vector \mathbf{h} and it is the responsibility of the optimizer to drive the residues to zero. This scheme is shown schematically in figure 3.2. However, this leads to an increase in computational expense at the optimizer level. It may be noted that it is possible to apply the concepts of analyzers and evaluators at system level as well as disciplinary level. This is elaborated in Chapter 4. Thus the manner in which the interdisciplinary coupling variables and coupling functions are treated and whether analysis or evaluators are used for solving (3.1), gives rise to various alternative formulations.

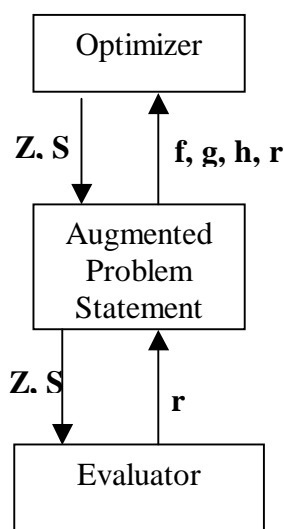


Fig.3.1 Schematic of an Engineering Design Problem using Evaluator

Alternative formulations should take into account features like

- Equivalence of formulation
- Ease of Implementation
- Disciplinary Autonomy
- Dimensionality

Equivalence of formulation refers to the fact that any solution of the alternative formulation must also be a solution of original problem statement (2.2).

If any of the analysis codes are modified, it should be possible to implement the same without having to modify other discipline codes. In other words, the changes should be ‘encapsulated’ within the discipline itself. This is referred to as Ease of Implementation.

One way of doing the MDA is to decompose the system of equations (3.1) along disciplinary lines like Aerodynamics, Propulsion, Structures etc. In such a case, the basic design vector is augmented by coupling variables. This offers the opportunity to carry out the various disciplinary analyses in parallel, without waiting for input from other disciplines.

Generally, the smaller the dimension of the problem, the better it is from the point of view of optimization. As seen earlier, performing ‘analysis’ does not introduce any additional variables or constraints. This results in minimum number of design variables. Decomposing the problem causes an increase in the size of the design variable vector. However, certain decomposition methods result in elimination of some design variables at the system-level, which may be practical advantage from the point of view of implementation. Another issue

related to dimensionality is that of the bandwidth of data exchange and the frequency with which the exchange occurs. It may be advantageous to exchange the data in the form of, say, coefficients of a spline fit, rather than in the original form. For example, the pressure data along flight vehicle body can spline fitted and used for exchanging information among the related disciplines, instead of directly sending the large number of original points.

As mentioned in the introduction, the desire to achieve computational autonomy of the disciplinary subsystems has led to many alternative MDO formulations. Formulations which attempt to cast the multidisciplinary system description as an optimization problem decomposed along organizational or physical characteristics of the problem, reflect what is known as the '**structural perspective**' [4]. In these types of formulations, the underlying idea is to consider the disciplines with as much autonomy as possible and coordinate the solutions of the disciplines at a system level. Solution of the system level problem is then attempted using any available optimization software. The physical structure of the problem thus dominates the formulation process. On the other hand, in the '**algorithmic perspective**' [4], the activity of formulation begins by considering the properties and abilities of the optimizer. The problem is then posed so that any conventional optimizer can be used to obtain reliable solutions. The underlying idea is to take in account the strong influence of the analytical features of problem formulations on the ability of the Non-linear programming algorithm to solve the problem reliably. The influence of the formulation is in terms of how many times the expensive analysis solutions have to be carried out during the optimization process and whether the optimizer can return a reliable solution for various formulations. Thus, in this type of problem formulation, the properties of the optimization algorithm assume greater importance rather than the physical structure of the problem. It may be noted that the distinction of 'structural perspective' and 'algorithmic perspective' is purely subjective. It is possible that same kind of formulation can arise when starting with either of the perspectives.

4. Classification of MDO Architectures

In this section, the basis for classification of MDO architectures is discussed. One way of classifying the formulations is based on whether the optimization is carried out at a single level or at a bi-level. In the single-level optimization approach, the task of determining the disciplinary design variables \mathbf{Z}_L and the system design variables \mathbf{Z}_S are entrusted to a single optimizer. In the bi-level approach, disciplinary optimizers are used to determine the disciplinary design variables while a system optimizer determines the system design variables.

As mentioned in the previous section, various types of MDO problem architectures are possible, based on the manner in which the inter-disciplinary coupling is treated and the manner in which MDA is carried out. There are two ways in which MDA can be done, namely: a) Simultaneous Analysis and Design (SAND) and (b) Nested Analysis and Design (NAND). In 'SAND', evaluators are used for computing the residues in the state equations at every optimization iteration. It is the responsibility of the optimizer to determine both the set of basic design variables and the augmented design variables, which drive the residue to zero. In NAND, the actual discipline analyzers are used to solve analysis equations and consequently there will not be any residue at solution. Based on these two distinctions, and whether a single level or a bi-level decomposition is done, it is possible to name the various formulations as follows:

- a) Single-NAND-NAND b) Single-SAND-SAND c) Single-SAND-NAND
- d) Bi-NAND-NAND e) Bi-SAND-SAND f) Bi-SAND-NAND

The first part of the name refers to whether the problem is posed as single level or a bi-level problem. The second part refers to the choice of analyzer/evaluator to treat the inter-disciplinary coupling and the third part refers to the choice of analyzer/evaluator at the discipline level solution of the disciplinary state equations. This type of classification is due to Balling [2]. It may be noted that certain combinations like NAND-SAND are obviously not possible.

The second method of classifying the architectures, suggested by Alexandrov[1], is based on the way the formulation handles the constraints. Constraints can be thought of as follows:

- a) Design Constraints (DC) : (**g** & **h**)

These can be specified at system level and at disciplinary level

b) Auxiliary Constraints:

These are of two types.

(i) Inter-disciplinary Consistency Constraints (ICC)- these are equality constraints which arise when disciplinary autonomy is introduced and the problem is decomposed along disciplinary lines to break the inter-disciplinary couplings and transfer the responsibility of ensuring inter-disciplinary consistency to the optimizer.

(ii) Disciplinary Analysis Constraints (DAC) – these are equality constraints which arise when the optimizer assumes the responsibility of carrying out MDA.

These distinctions are illustrated through a three discipline MDO problem, posed as below:

Find \mathbf{Z} which

Minimize $f(\mathbf{Z}, \mathbf{S})$

Subject to $\mathbf{g}_0(\mathbf{Z}, \mathbf{S}) \leq 0$

$$\mathbf{g}_1(\mathbf{z}_1, \mathbf{s}_1) \leq 0$$

$$\mathbf{g}_2(\mathbf{z}_2, \mathbf{s}_2) \leq 0$$

$$\mathbf{g}_3(\mathbf{z}_3, \mathbf{s}_3) \leq 0 \quad (4.1)$$

where $\mathbf{S} = \mathbf{s}_1 \cup \mathbf{s}_2 \cup \mathbf{s}_3$ and $\mathbf{Z} = \mathbf{z}_1 \cup \mathbf{z}_2 \cup \mathbf{z}_3$ and $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$ denotes the solution of the multidisciplinary analysis

$$A_1(\mathbf{z}_1, \mathbf{s}_1, \mathbf{y}_{21}, \mathbf{y}_{31}) = 0 \quad (4.2)$$

$$A_2(\mathbf{z}_2, \mathbf{s}_2, \mathbf{y}_{12}, \mathbf{y}_{32}) = 0 \quad (4.3)$$

$$A_3(\mathbf{z}_3, \mathbf{s}_3, \mathbf{y}_{13}, \mathbf{y}_{23}) = 0 \quad (4.4)$$

\mathbf{g}_0 is the system level constraint vector. \mathbf{g}_1 and \mathbf{g}_2 are disciplinary constraints. These constraints together represent the Design Constraints. Equations 4.2 –4.4, can be alternatively written as follows:

$$A_1(\mathbf{z}_1, \mathbf{s}_1, \mathbf{y}^*_{21}, \mathbf{y}^*_{31}) = 0 \quad (4.5)$$

$$A_2(\mathbf{z}_2, \mathbf{s}_2, \mathbf{y}^*_{12}, \mathbf{y}^*_{32}) = 0 \quad (4.6)$$

$$A_3(\mathbf{z}_3, \mathbf{s}_3, \mathbf{y}^*_{13}, \mathbf{y}^*_{23}) = 0 \quad (4.7)$$

where \mathbf{y}^*_{21} , \mathbf{y}^*_{31} , \mathbf{y}^*_{12} , \mathbf{y}^*_{32} , \mathbf{y}^*_{13} and \mathbf{y}^*_{23} are coupling variables vectors which are introduced to make the disciplines independent and act as inputs for disciplines 1, 2 and 3 respectively. However, to ensure consistency of the multi-disciplinary analysis, it is then required to introduce Interdisciplinary Consistency Constraints (ICC) of the following form:

$$\mathbf{y}^*_{21} - \mathbf{y}_{21} = 0, \mathbf{y}^*_{31} - \mathbf{y}_{31} = 0,$$

$$\mathbf{y}^*_{12} - \mathbf{y}_{12} = 0, \mathbf{y}^*_{32} - \mathbf{y}_{32} = 0,$$

$$\mathbf{y}^*_{13} - \mathbf{y}_{13} = 0, \mathbf{y}^*_{23} - \mathbf{y}_{23} = 0$$

State equations 4.4 - 4.6 or their alternative form 4.5 – 4.7, represent Disciplinary Analysis Constraints (DAC).

Having noted the nature of the constraints, we say that MDO formulation is ‘closed’ with respect to constraints if the formulation, rather than the optimizer, shoulders the responsibility of satisfying the constraints through analysis. Use of evaluators, instead of analyzers, makes the formulation rely on the optimizer to enforce the constraints, and the formulation is termed as ‘open’. So based on the manner, in which the above constraints are evaluated, it is possible to assign to each of constraints the status of being ‘closed’ or ‘open’. It may be noted that ‘SAND’ can be interpreted as a parallel terminology for ‘open’, while ‘NAND’ can be interpreted as a parallel terminology for ‘closed’.

5. Single level Formulations

In this chapter, we discuss three single level formulations [2,5] and compare their features. The formulations are explained with respect to a three discipline coupled system.

5.1 Analytical Features and Mathematical Statement

Single NAND-NAND

In Single-NAND-NAND formulation, the optimizer controls only the basic design variable vector, \mathbf{Z} . In terms of the number of variables, this formulation has the minimum. For every set of design variables sent by the optimizer, complete MDA is performed using the specialized disciplinary analysis tools. The solution vector, ie., the state variables, used for evaluation of the constraints and objective function are returned to the optimizer. Since MDA is performed at every iteration step, one is assured of a complete interdisciplinary feasible solution. As such, no auxiliary constraints need to be introduced and the constraint set is given below.

Design Constraints (DC)

$$\mathbf{g}_0 \leq 0 \quad (\text{system design constraint})$$

$$\mathbf{g}_1 \leq 0 ; \mathbf{g}_2 \leq 0 ; \mathbf{g}_3 \leq 0 \quad (\text{disciplinary design constraint})$$

Optimization problem statement:

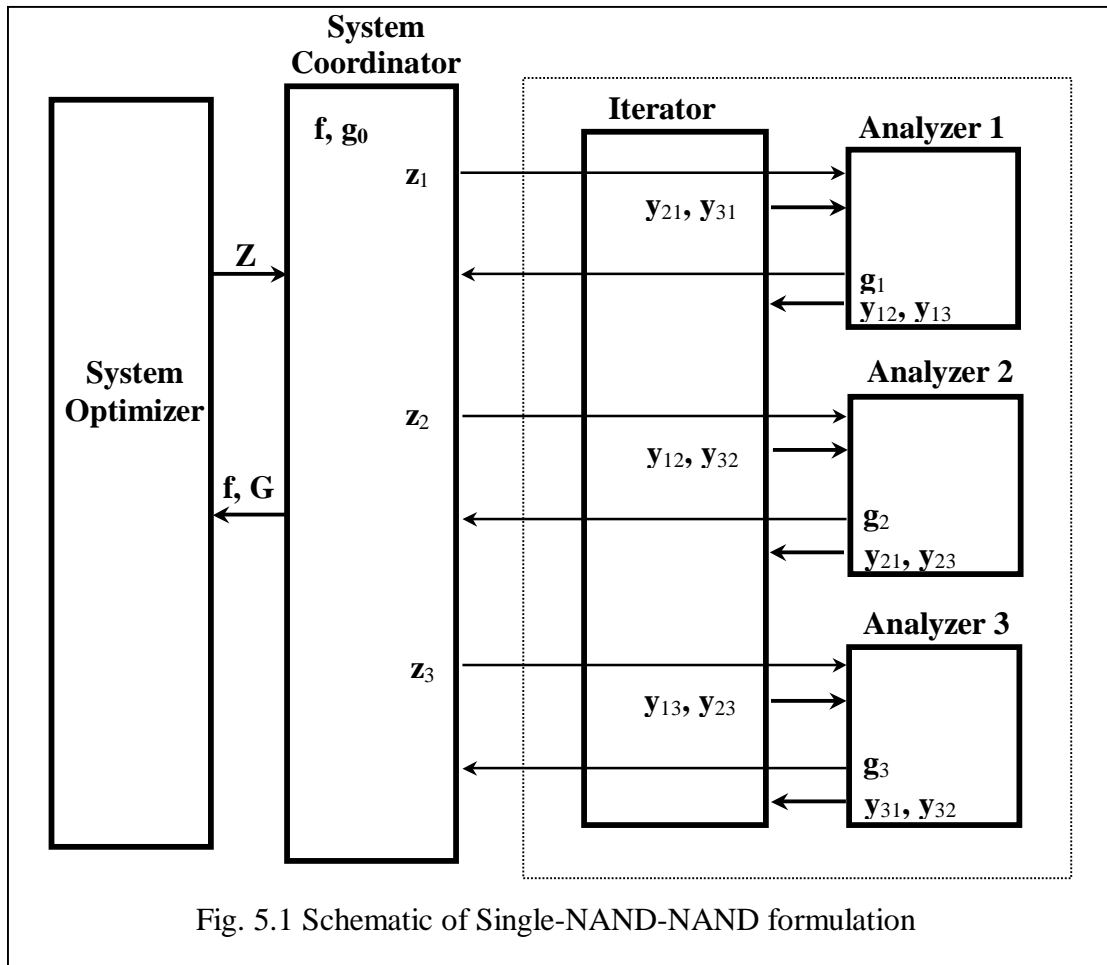
Find \mathbf{Z} which

Minimize $f(\mathbf{Z})$

Subject to

Dc's as stated above.

This is pictorially shown in figure. 5.1. In the figure \mathbf{G} represents all the constraints. This formulation has been widely referred to as Multi-Discipline Feasible (MDF) approach. Note that the word feasible refers to getting converged solution for the MDA. It does not refer to the design constraints being satisfied. Following Alexandrov's notation, this formulation can be classified as "closed" with respect to disciplinary analysis and interdisciplinary consistency constraints and "open" with respect to disciplinary design constraints. Hence, we can denote the formulation as CDAC/ODC/CICC. This formulation reflects the 'algorithmic perspective'. Mathematically the problem is well-posed and any standard Non-Linear Programming algorithm can be used to solve problems cast in this formulation.



Single-SAND-SAND

In this approach, discipline evaluators, rather than analyzers, are used to evaluate the interdisciplinary feasibility as well as the disciplinary state equations. As a result the basic design vector is augmented by the inclusion of coupling variables and state variables and the constraint set is augmented by auxiliary constraints. A single optimizer controls the augmented vector of design variables. The mathematical description of this form is as given below:

Augmented Design Variable Vector:

$$\mathbf{Z}_{\text{aug}} = (\mathbf{Z}, \mathbf{S}, \mathbf{y}^*_{12}, \mathbf{y}^*_{13}, \mathbf{y}^*_{21}, \mathbf{y}^*_{23}, \mathbf{y}^*_{31}, \mathbf{y}^*_{32})$$

Design Constraints (DC):

$$\mathbf{g}_0 \leq 0 \quad (\text{system design constraints})$$

$$\mathbf{g}_1 \leq 0 ; \mathbf{g}_2 \leq 0 ; \mathbf{g}_3 \leq 0 \quad (\text{disciplinary design constraints})$$

Auxiliary Constraints:

$$\left. \begin{aligned} \mathbf{y}_{21} - \mathbf{y}^*_{21} = 0 ; \mathbf{y}_{31} - \mathbf{y}^*_{31} = 0 \\ \mathbf{y}_{12} - \mathbf{y}^*_{12} = 0 ; \mathbf{y}_{32} - \mathbf{y}^*_{32} = 0 \end{aligned} \right\} \quad (\text{ICC's})$$

$$\mathbf{y}_{13} - \mathbf{y}^*_{13} = 0; \mathbf{y}_{23} - \mathbf{y}^*_{23} = 0$$

$$\begin{aligned} \mathbf{r}_1 &= \mathbf{s}_1 - \mathbf{E}_1(\mathbf{z}_1, \mathbf{y}^*_{21}, \mathbf{y}^*_{31}) = 0 \\ \mathbf{r}_2 &= \mathbf{s}_2 - \mathbf{E}_2(\mathbf{z}_2, \mathbf{y}^*_{12}, \mathbf{y}^*_{32}) = 0 \quad (\text{DAC's}) \\ \mathbf{r}_3 &= \mathbf{s}_3 - \mathbf{E}_3(\mathbf{z}_3, \mathbf{y}^*_{13}, \mathbf{y}^*_{23}) = 0 \end{aligned}$$

Optimization problem statement:

Find \mathbf{Z}_{aug} which

Minimize $f(\mathbf{Z}_{\text{aug}})$

Subject to

DC's, ICC's and DAC's as stated above.

The above statement is pictorially shown in figure 5.2. In the figure, \mathbf{G} and \mathbf{R} represent all the constraints \mathbf{g} and residues \mathbf{r} , respectively.

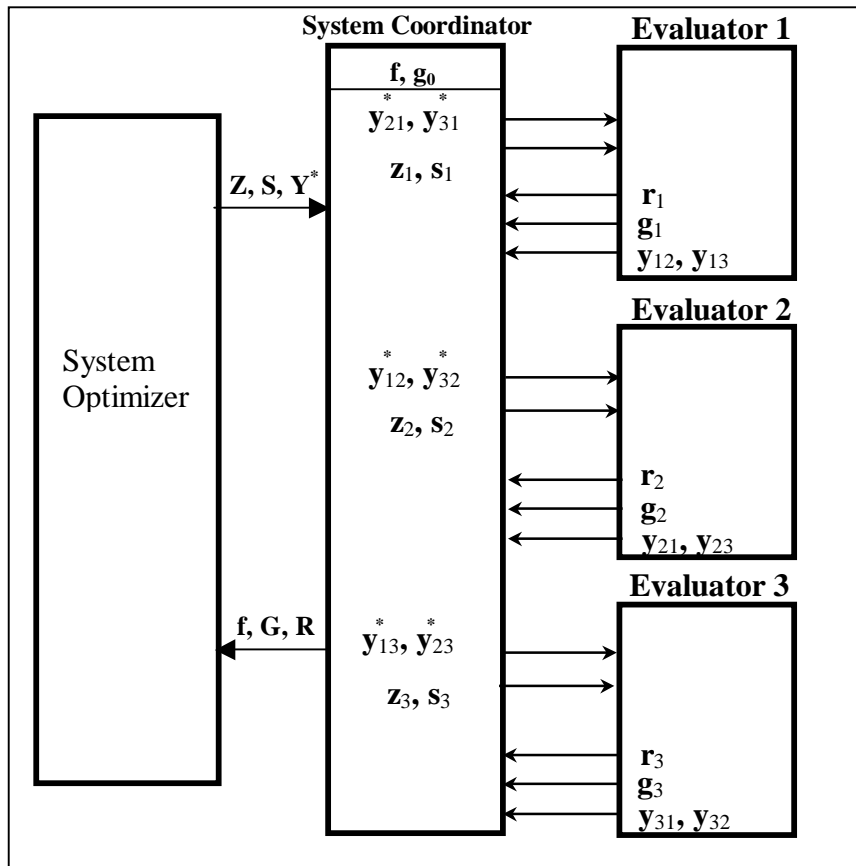


Fig. 5.2 Schematic of Single-SAND-SAND formulation

The motivation for this type of formulation is that a complete MDA consistent solution is not required when one is far away from the optimal solution. The residues are progressively reduced to zero as one approaches the optimal solution. Thus, the computational cost may be reduced as compared to performing full MDA, provided the optimizer is able to handle the augmented optimization problem load. However, if the optimization process is terminated for some reason, one does not have a discipline feasible solution, let alone a MDA solution. This formulation has also been referred to as the “All-at-Once”, since the design and optimization occur simultaneously. Following the notation of Alexandrov, the formulation can be denoted as open analysis/open design constraints/open interdisciplinary constraints (ODAC/ODC/OICC).

Single SAND-NAND

This formulation is similar to that of Single-SAND-SAND, except that disciplinary analyzers rather than the disciplinary evaluators are used to solve the discipline analysis equations. The state variables are not required to be included in the design vector. Output information from other disciplines which is required as input for a discipline analysis, is represented through coupling variables so that it is possible to carry out complete discipline level analysis. Thus the design vector is augmented by inclusion of only coupling variables and is given as below:

Augmented Design Variable Vector:

$$\mathbf{Z}_{\text{aug}} = (\mathbf{Z}, \mathbf{y}^*_{12}, \mathbf{y}^*_{13}, \mathbf{y}^*_{21}, \mathbf{y}^*_{23}, \mathbf{y}^*_{31}, \mathbf{y}^*_{32})$$

It may be noted that the solutions returned from the disciplines are individually consistent and as such residues will not be there. However, the inter-disciplinary consistency is not achieved. Ensuring the task of inter-disciplinary consistency is the responsibility of the optimizer. The design constraints and the auxiliary constraints are given as below:

Design Constraints (DC):

$$\mathbf{g}_0 \leq 0 \quad (\text{system design constraints})$$

$$\mathbf{g}_1 \leq 0 ; \mathbf{g}_2 \leq 0 ; \mathbf{g}_3 \leq 0 \quad (\text{disciplinary design constraints})$$

Auxiliary Constraints:

$$\left. \begin{aligned} \mathbf{y}_{21} - \mathbf{y}_{21}^* = 0; \mathbf{y}_{31} - \mathbf{y}_{31}^* = 0 \\ \mathbf{y}_{12} - \mathbf{y}_{12}^* = 0; \mathbf{y}_{32} - \mathbf{y}_{32}^* = 0 \\ \mathbf{y}_{13} - \mathbf{y}_{13}^* = 0; \mathbf{y}_{23} - \mathbf{y}_{23}^* = 0 \end{aligned} \right\} \quad (\text{ICC's})$$

Optimization problem statement:

Find \mathbf{Z}_{aug} , which

Minimize $f(\mathbf{Z}_{\text{aug}})$

Subject to

DC's and ICC's as stated above.

The pictorial representation of the mathematical statement is shown in figure 5.3. In the figure, \mathbf{G} represents all the constraints \mathbf{g} .

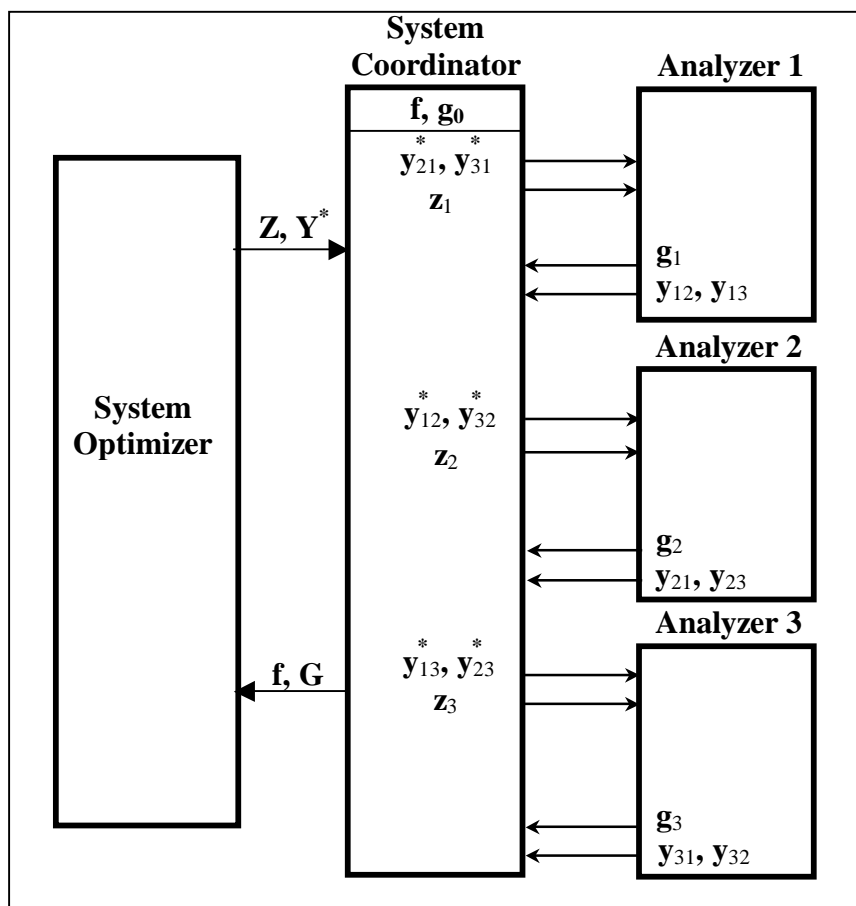


Fig.5.3 Schematic of Single-SAND-NAND formulation

This approach reflects an “in-between” position compared to the SAND-SAND and NAND-NAND approaches. It has been alternatively referred to in the literature as

“Individual Discipline Feasible (IDF)” [5] and “Distributed Analysis Optimization” [4]. Again, feasibility here refers to consistency in discipline solution and not to qualification of the design constraints. Following Alexandrov’s notation, the formulation is said to be “closed” with respect to disciplinary analyses and “open” with respect to design and inter-disciplinary constraints (CDAC/ODC/OICC). The nature of the architecture offers scope to use the existing specialized disciplinary analysis codes with little modifications.

5.2 Comparative features of Single level formulations

MDF or NAND-NAND approach has the fewest number of design variables and is easy to comprehend. IDF or SAND-NAND is augmented by inclusion of coupling variables, which are needed to ensure inter-disciplinary compatibility. All-at-Once or SAND-SAND has the largest number of design variables, due to the introduction of coupling variables and state variables in the optimization design vector.

All-at-Once and MDF architectures represent two extreme strategies of solving the Multidisciplinary Analysis problem. While All-at-once formulation does not require feasibility or consistency of solution till the point of optimality, the MDF formulation requires complete feasible solution of the coupled system at each iteration of optimization. Thus, if high fidelity tools like CFD are used for disciplinary analysis, implementation in MDF formulation may have practical problems. All-at-Once provides most efficient means towards getting a solution, by not enforcing the requirement of feasibility at every iteration. But, again, if the problem on hand is of large scale or if the problem is tightly coupled, then the number of iterations to achieve convergence may be very large. This negates the advantage of not having to perform complete MDA. In IDF formulation, it is required to satisfy feasibility of solution with respect to only the individual disciplines. Ensuring interdisciplinary consistency becomes the responsibility of the optimizer.

Thus the formulation to be chosen is very much dependant on the nature of the problem to be solved. However IDF formulation, which reflects an “in-between” strategy compared to the two extreme strategies adopted by MDF and All-at-Once, may hold promise for solving industry type problems. In fact, one of upcoming formulations namely Collaborative Optimization (discussed in Ch 6), can be thought

of as an advanced concept of IDF, wherein in addition to discipline solution feasibility it is also required to maintain feasibility of discipline specific constraints.

6.Bi-Level Formulations

In this chapter, we discuss the motivation for bi-level formulations, followed by description of two architectures namely, Collaborative Optimization Architecture [2] and Concurrent Subspace Optimization [7]. There are other bi-level architectures like BLISS [8] and other variations in Collaborative formulations [9], as well.

As mentioned in Chapter 1, research in the area of MDO has been active for the past decade and several types of formulations have emerged. However, none of them has yet been demonstrated on an industry level design problem. A few instances, where MDO approach has been used, focussed on the development of large multidisciplinary synthesis programs. Generally in these programs, the discipline analysis is based on low-fidelity or simple models to solve the design problem. The effect of high fidelity analysis and their implementation is yet to be studied. Some of the reasons for MDO formulations not being used in the industry at present are seen to be as follows:

- ‘*a-priori*’ time investment needed to integrate large design problems
- Industry prefers to use proven design practices. Using new design paradigms in project environment is considered risky.

Spurred by these facts, research on new types of formulations like Collaborative Optimization (CO) and Concurrent Sub-Space Optimization (CSSO), were launched.

Stanford University has been studying CO and University of Notre-Dame, the CSSO. The nature of these formulations is believed to be very much analogous to the design environment prevalent in industry.

Three generic types of formulations in a Bi-level architecture are possible:

- SAND-SAND
- SAND-NAND
- NAND-NAND

As indicated earlier, the word 'Bi' refers to the fact that separate optimizers are used at system and discipline levels. SAND/NAND indicates the type of analysis performed ie, whether analyzers or evaluators are used at each level. CO can be thought of SAND-NAND type of formulation, while CSSO can be thought of as SAND-SAND type of formulation.

6.1 Collaborative Optimization Architecture

6.1.1. Introduction

Collaborative Architecture (CO) [3] is developed with a motivation to include industry design perspective in the formulation characteristics. In industry environment, disciplinary experts are allowed to contribute to the overall decision-making process, though the final authority is vested with the project leader. CO architecture reflects this 'structural perspective'.

The complex MDA problem is hierarchically decomposed along disciplinary analysis boundaries. Each discipline/sub-space has associated with it an optimizer, which controls design variables local to the discipline. Outputs of other disciplines required as input for analysis, are treated as auxiliary design variables in the subspace optimization. Thus, each sub-space has freedom to choose its own value for the interdisciplinary variable. For each of such inter-disciplinary variables, system optimizer issues system target variable to sub-spaces. Within the sub-space, these system level targets are treated as fixed parameters. The objective of the sub-space /discipline optimizer is to minimize the discrepancy between the system level target and the local value for the inter-disciplinary variable. In the basic architecture, this discrepancy is modeled in a least square sense. The sub-space optimization problem is also charged with the task of satisfying the discipline specific constraints. The system optimizer coordinates all the subspace optimization problems and ensures interdisciplinary consistency. This is enforced by requiring each sub-space objective

(discrepancy function) to be zero through an equality constraint at the system level. Thus, there will be as many equality constraints as there are subspaces. Since, at each iteration of system level optimization, the discipline constraints are satisfied, the architecture is also referred to as disciplinary constraint feasible method. Following Alexandrov's method of classification at subspace optimization level, this formulation is 'closed' with respect to discipline analysis and design constraints and 'open' with respect to interdisciplinary constraints (CDAC/CDC/OICC). A schematic representation of C-O architecture is shown in figure 6.1. Notations used are explained in the next section.

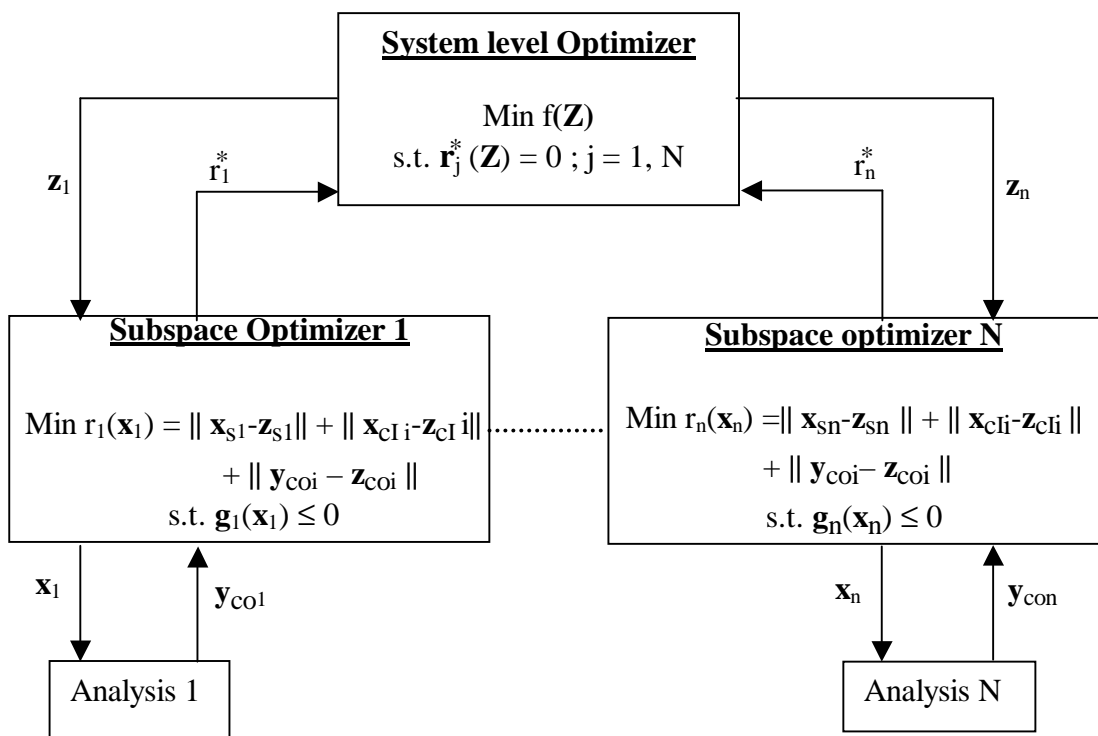


Figure 6.1 Schematic of Collaborative Optimization

6.1.2 Mathematical Formulation in CO

Decomposition of the original problem

The original problem defined in Ch3 is stated again

$$\text{Minimize } f(\mathbf{Z}, \mathbf{S}(\mathbf{Z}))$$

$$\text{Subject to } \mathbf{g}(\mathbf{Z}, \mathbf{S}(\mathbf{Z})) \leq 0 ; \mathbf{h}(\mathbf{Z}, \mathbf{S}(\mathbf{Z})) = 0 \quad (2.2)$$

Where ‘S’ is a state variable vector and is obtained by solving state equations of the form

$$\mathbf{A}(\mathbf{Z}, \mathbf{S}(\mathbf{Z})) = 0 \quad (2.3)$$

In single level formulations the control over the design vector \mathbf{Z} is with one optimizer i.e., the system optimizer. There are no optimizers at disciplinary level and the role of discipline is restricted only to analysis. In CO, the disciplines are given autonomy or control over their respective disciplinary design variables through the disciplinary optimizer. Hence some of the variables can be transferred to the disciplines and the disciplinary design variables need to be denoted distinctly from the system level variables. Recall from Chapter 2, that the system level design vector consists of

- i) \mathbf{Z}_L : representing a union of local variables \mathbf{z}_{Li} used for i^{th} disciplinary analysis or evaluation
- ii) \mathbf{Z}_S : representing design variables shared by two or more disciplines
- iii) \mathbf{y}^*_{ij} and \mathbf{y}_{ij} representing the auxiliary or coupling variables and functions

In CO, control over the local variables \mathbf{z}_{Li} is exercised by the disciplinary optimizers and these are no longer a part of the system level design variable vector. Thus for CO, the system level vector is $\mathbf{Z}_{\text{aug}} = \mathbf{Z}_S \cup \mathbf{Z}_c$, where $\mathbf{Z}_c (= \cup \mathbf{z}_{ci})$ is an alternate compact notation for the auxiliary vector representing the union of coupling variables and functions. \mathbf{z}_{ci} can further be expressed as $\mathbf{z}_{ci} = \mathbf{z}_{cli} + \mathbf{z}_{coi}$, where $\mathbf{z}_{cli} = \cup \mathbf{y}^*_{ij}$ and $\mathbf{z}_{coi} = \cup \mathbf{y}_{ij}$. \mathbf{z}_{coi} denotes output of discipline ‘i’ required by any other discipline $j \neq i$. Similarly, \mathbf{z}_{cli} denotes input to discipline ‘i’ from any other discipline ‘j’. The system level variables $\mathbf{z}_{i\text{aug}} = \mathbf{z}_{si} \cup \mathbf{z}_{ci}$, are sent to the i^{th} discipline as system level targets. Let the disciplinary level variable vector for i^{th} discipline be designated as \mathbf{x}_i . It consists of the following components:

$$\mathbf{x}_i = \mathbf{x}_{Li} \cup \mathbf{x}_{si} \cup \mathbf{x}_{cli} \cup \mathbf{x}_{coi}$$

where ,

\mathbf{x}_{Li} is the local design variable vector ($\mathbf{X}_L = \cup \mathbf{x}_{Li}$)

\mathbf{x}_{si} is the copy of the vector \mathbf{z}_{si}

\mathbf{x}_{cli} is the copy of the vector \mathbf{z}_{cli}

\mathbf{x}_{coi} is the copy of the vector \mathbf{z}_{coi}

Similarly, the constraint vectors \mathbf{g} and \mathbf{h} are also partitioned and transferred to the respective disciplines.

System level Optimization Problem

The system level optimization problem is stated as follows:

Find \mathbf{Z}_{aug} , which

$$\begin{aligned} \text{Min. } & F(\mathbf{Z}_S) \\ \text{s.t. } & \mathbf{r}^*(\mathbf{Z}_{\text{aug}}) = 0 \end{aligned} \quad (6.1)$$

where ,

F : system-level objective function

\mathbf{Z}_{aug} = system-level design variable vector (targets issued to sub-spaces)

\mathbf{r}^* : system-level non-linear constraint vector, whose elements are discrepancy functions returned from solution of the sub-space optimization problems

The system-level solution is defined as,

$$F = F^{**}, \mathbf{Z} = \mathbf{Z}^{**} \text{ and } \mathbf{X}_L = \mathbf{X}_L^{**}$$

Discipline / Subspace Optimization Problem

For a 'n' discipline problem, there will be 'n' sub-space optimization problems. The design vector and the mathematical statement for an i^{th} sub-space is as stated below.

Find \mathbf{x}_i , which

$$\begin{aligned} \text{Min. } & r_i(\mathbf{x}_i) = \|\mathbf{x}_{si} - \mathbf{z}_{si}\| + \|\mathbf{x}_{cli} - \mathbf{z}_{cli}\| + \|\mathbf{y}_{cli} - \mathbf{z}_{coi}\| \\ \text{Subject to } & \mathbf{g}_i(\mathbf{x}_i) \leq 0 ; \mathbf{h}_i(\mathbf{x}_i) \leq 0 \end{aligned} \quad (6.2)$$

The sub-space solution is defined as

$$r_i = r_i^* ; \mathbf{x}_i = \mathbf{x}_i^*$$

The norm in the objective function $r_i(\mathbf{x}_i)$ is generally, calculated as L_2 norm.

6.1.3 Collaborative Solution Process

The collaborative solution process starts with an initial guess for the system level targets \mathbf{Z}_{aug} at the zeroth iteration. Each subspace receives the required elements

of the system level target vector relevant to it. These are treated as fixed parameters in the sub-space optimization problem. During the solution of the subspace optimization, the interdisciplinary design variables \mathbf{x}_s and \mathbf{x}_c move as close as possible towards their target value \mathbf{z}_s and \mathbf{z}_c , while maintaining feasibility with respect to sub-space constraints $\mathbf{g}(\mathbf{x})$. A characteristic of the formulation is that each subspace is allowed freedom to choose its own value for the interdisciplinary variable. This freedom enables the subspace problem to satisfy the discipline specific constraints. Optimum values of the discrepancy function r^* (ie, the objective function of the subspace) is returned to the system level for evaluating the system equality constraints. Thus, the system optimizer shoulders the responsibility of interdisciplinary consistency. The system level issues a fresh target \mathbf{Z}_{aug} , such that there is an improvement in the system objective function. The process is repeated until system convergence, $\mathbf{Z} = \mathbf{Z}^{**}$, is attained.

6.1.4 Sensitivity Analysis

Sensitivity analysis refers to studying the variation of the optimal objective function due to perturbations in the parameters or constraint limits. The post optimality equation derived in [3] is described briefly below:

Consider a general problem

Min. $F(\mathbf{z})$

Subject to $\mathbf{g}(\mathbf{z}, \mathbf{p}) \leq 0$

At the point of optimality we can write the objective function $F^* = F^* + \lambda^T \mathbf{g}^*$, since $\mathbf{g}^* = 0$, as only active constraints need to be considered for sensitivity analysis.

Taking the total derivative of F^* with respect to parameter \mathbf{p} , we have

$$dF^* / d\mathbf{p} = (\partial F^* / \partial \mathbf{p} + \lambda^T \partial \mathbf{g}^*(\mathbf{z}^*, \mathbf{p}) / \partial \mathbf{p}) + (\partial \mathbf{z}^* / \partial \mathbf{p}) \partial / \partial \mathbf{z}^* (F^* + \lambda^T \mathbf{g}^*(\mathbf{z}^*, \mathbf{p}))$$

From the first order necessary condition

$$\partial / \partial \mathbf{z}^* (F^* + \lambda^T \mathbf{g}^*(\mathbf{z}^*, \mathbf{p})) = 0$$

Hence, irrespective of the value of $(\partial \mathbf{z}^* / \partial p_j)$, we have

$$dF^* / d\mathbf{p} = \partial F^* / \partial p_j + \lambda^T \partial \mathbf{g}^*(\mathbf{x}^*, \mathbf{p}) / \partial p_j \quad (6.3)$$

Equation (6.3) is referred to as the *first-order post optimality equation*. $dF^* / d\mathbf{p}$ represents the change in the optimum objective function value with respect to a parameter variation. \mathbf{z}^* can change in the process, and second order information is

needed to estimate the new \mathbf{z}^* . The details are provided in [3]. $\partial F^* / \partial p_j$ represents the derivative of the objective function evaluated at the optimum, \mathbf{z}^* .

The nature of CO formulation allows development of analytic expressions for the system level constraint jacobian. This is possible due to the fact that the system level variable is treated as a parameter at the subspace level.

6.1.5 Problems in C-O Architecture

Two problems have been reported in the basic CO formulation [3,6]. One is the non-existence of Lagrange multipliers as the system level solution approaches the optimal point and the other is the non-smoothness of the system level constraint Jacobian matrix.

Non-Existence of Lagrange Multiplier

The basic C-O formulation at system level is repeated below:

$$\text{Min } F(\mathbf{Z}_s)$$

$$\text{s.t. } r^*_j(\mathbf{Z}) = 0 \quad j = 1, n$$

$$\text{where, } r^*_j = \sum (x_s - z_s)^2 + \sum (x_{cl} - z_{cl})^2 + \sum (y_{co} - z_{co})^2 .$$

r^* is the objective function of the j^{th} sub-space optimization problem. Note that this is not a part of the original problem defined by (2.2) and is a result of decomposition of problem. *For ease of explanation let \mathbf{x} denote all the types of interdisciplinary variables and \mathbf{z} the corresponding target variables.* The Lagrangian at the system level can then written as

$$L = F^* + \sum \lambda_j (x_j - z_j)^2$$

Most gradient-based optimization algorithms require that the first order necessary conditions be satisfied at the point of optimality.

$$\text{Thus } \nabla_{\mathbf{z}} L = 0$$

$$\nabla_{\mathbf{z}} F^* + \lambda_j \nabla_{\mathbf{z}} r^*_j = 0 \quad (6.4)$$

The above equation implies that system level constraint jacobian, $\nabla_{\mathbf{z}} r^*_j$, must be defined for existence of the Lagrange multiplier λ_j . Recall that r^*_j is the sub-space objective function. z_j is a system target issued to the subspace 'j' and is treated a fixed parameter. Making use of the post optimality equation (6.3) with ' r^* ' as the function and ' z_j ' as the parameter we have the following:

$$dr^*_j / dz_j = \partial r^* / \partial z_j + \lambda^T \partial g(x_j) / \partial z_j$$

Since 'g' is a domain specific constraint and is not a function of \mathbf{z} , we have

$\partial g(x_j) / \partial z_j = 0$. Hence the system level gradient is

$$dr_j^* / dz = \partial r_j^* / \partial z_j = -2(x_j - z_j) \quad (6.5)$$

As the system level solution approaches, we have

$x_j \rightarrow z_j$ and so from (6.5),

$$\partial r^* / \partial z_j = 0$$

The above equation results in non-existence of Lagrange multipliers in equation (6.4).

To overcome this problem, an alternative formulation was suggested.

Alternative Formulation [3]

The basic system level problem is refined as follows:

Min $F(\mathbf{Z}_s)$

$$\text{Subject to } \mathbf{r}(\mathbf{z}) = \mathbf{x} - \mathbf{z} = 0 \quad (6.6)$$

Where,

\mathbf{r} : system level constraint returned from the solution of the subspace optimization problem

In this refinement, instead of 'n' cumulative constraints, one for each sub-space problem, there are as many constraints as there are inter-disciplinary variables. Also, the system level constraint is no longer the subspace objective function as in the basic formulation. Invoking again the post-optimality equation (6.3), we have

$$d\mathbf{r} / d\mathbf{z} = d\mathbf{x} / d\mathbf{z} - \delta_{jk}$$

$$\delta_{jk} = 0 \quad j \neq k$$

$$1 \quad \text{for } j = k$$

and $d\mathbf{x} / d\mathbf{z}$ can be calculated from the second order sensitivity information.

Thus in the refined formulation, as the solution approaches optimality the system level constraint jacobian $d\mathbf{r} / d\mathbf{z}$ is defined and as a result the problem of non-existence of Lagrangian multiplier in equation (6.4) is avoided.

Non-smoothness of system level constraint jacobian

A basic requirement of solving non-linear programming problem using a gradient based optimizer, is that the functions evaluated by the disciplinary analyses must be smooth. The disciplinary experts are generally required to ensure this. In CO

architecture, the original problem is decomposed into discipline based sub-space optimization problems and a system level problem. Sub-space objective functions are thus introduced in the transformed problem statement and it is required to return ‘ \mathbf{r}^* ’ to the system level problem. Note that these function calls are not a part of the original problem statement. Thus utmost care must be exercised while defining the sub-space objective function, so that it returns a smooth solution to the system level. In the basic CO formulation, the discrepancy between the inter-disciplinary variable and the associated system level target is the objective function and this must be driven to zero. However, the inter-disciplinary output is a solution of non-linear function and can have multiple solutions. As a result, when the system parameter \mathbf{z} is varied, the solution ‘ \mathbf{r}^* ’ can jump from one solution to another. This makes the function, non-smooth. Consequently, computing the system constraint jacobian, becomes difficult. A detailed numerical illustration is provided in [3]. It is briefly described below:

A non-linear sub-space problem can be formulated as

$$\text{Min } r_1 = (x_s - z_s)^2 + (y_{co} - z_{co})^2$$

$$0 \leq x_s \leq 1 \text{ and } 0 \leq x_L \leq 1$$

Where $y_{co} = 100(x_L - x_s^2)^2$, represents an inter-disciplinary output variable.

As the system level solution approaches, we have

$$y_{co}^* = z_{co}; x_s = z_s \text{ and } r_1^* = 0;$$

$$\text{Therefore, } x_L = z_s^2 + (z_{co}/100)^{1/2}$$

The above equation has two roots. The solution can jump from one root to another, when the system level target variable z_{co} varies. Details of circumventing the above problem are provided in [3].

6.2 Concurrent Subspace Optimization Formulation

Concurrent subspace optimization is another bi-level formulation which is being developed at University of Notre-Dame [7]. Unlike in CO, no specific distinction is made in the choice of objective function for system level problem and the sub-space level problem. The system level problem is treated just as another discipline and each discipline has its own optimizer.

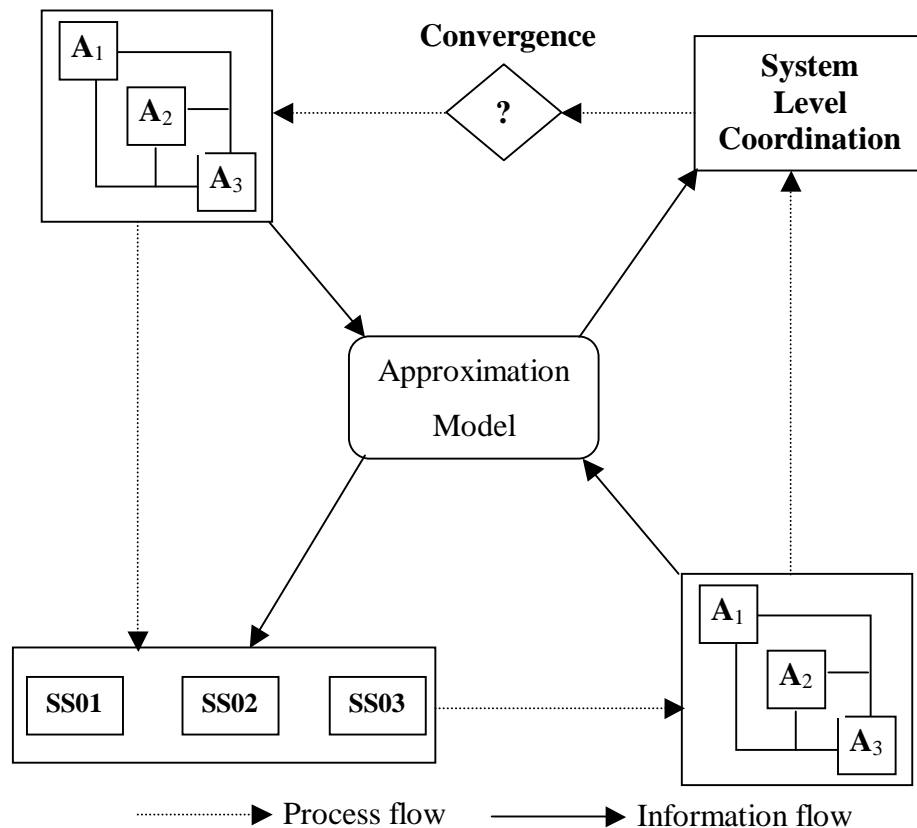


Fig.6.2 Schematic of CSSO formulation

The system level optimizer co-ordinates the subspace optimization problems and ensures that all the constraints are satisfied at convergence. The formulation is briefly described in this section. There are mainly four processes in the CSSO formulation and these are schematically depicted in figure 6.2. The solution process proceeds in counterclockwise direction, starting with system analysis (in the upper left hand corner). The system analysis encompasses within it, the task of solving the coupled set of disciplinary analyses for a given set of initial designs. Solving the above equations requires an iterative method. This process is referred to as System Analysis. On solution, we get a consistent set of states, which describe the behavior of the system. However, the solution need not satisfy the constraints of the optimization

problem. The purpose of solving the coupled system is to create data, using which approximate models for the system behavior can be constructed. The model can then be used by the disciplinary experts to understand the influence of local design decisions on the system level performance. All the subspace optimization problems are entrusted with task of minimizing the same system objective function. For states, which are local to it, the sub-space performs detailed analysis while for the non-local states it makes use of approximate models. This constitutes the second process. The designs returned by the sub-space solutions are used to augment the database and create progressively better approximate models. This constitutes the third process. Using the approximate models of the sub-spaces, the system level optimization problem is solved. This constitutes the fourth and final process in the CSSO formulation. The processes are repeated until convergence.

6.3 Comparison of CO and CSSO

The strategies of CO and CSSO mainly differ in the manner in which the inter-disciplinary consistency is achieved. In CO, the sub-problems are entrusted with the task of minimizing the discrepancy functions and these are augmented as equality constraints at system level optimization problem. In CSSO formulation, all the sub-problems are given the task of satisfying the same objective function ie. the system objective function. For information on non-local states, the sub-problems use approximate models. In CO the sub-problems are given the freedom to disagree on the value of the inter-disciplinary variable. Driving the discrepancy, between the inter-disciplinary variable and the associated system level target, to zero ensures consistency. Thus, the size of the design problem will be more in CO than in CSSO. The nature of the decomposition followed in the basic architecture of CO results in problems to the optimizer, described in the previous section. CSSO formulation does not have any such inherent problems. However, CSSO formulation requires one to construct approximate models, which may add to overhead computational cost. Also, the accuracy of the approximate model has to be good. In CO, the system level sensitivity gradients can be obtained through analytical means. In CSSO, one has to rely on finite-difference, which requires tight convergence.

7. Computational Performance of Two Architectures

In this section, the computational performance of Collaborative Optimization with that of a single level formulation, namely All-at-Once is discussed briefly. The criteria for the comparative study of the performance chosen in [10], is the number of function evaluations required to find an optimal solution. Other methods of comparison of computational performance are given in [2] and [9].

Assumptions

The optimization algorithm used and the nature of the non-linearity of the problem strongly influence the number of function evaluations. It is assumed that all the gradient information is accurately computed by forward difference method, and the non-linearity is quadratic and the optimizer is a sequential quasi-Newton solver. Then the following hold:

- The number of analysis evaluations required to find a line search direction is one plus the number of inputs to the analysis. ‘One’ refers to the baseline evaluation. Rest of the evaluations, are for computing the gradients of the function with respect to each design variable.
- Maximum number of line searches required to find an optimum is equal to the total number of design variables.

Recall that in All-at-Once formulations it is required to solve an ‘n’ discipline system of equations using evaluators. The maximum possible total number of function evaluations to determine a stationary point for the j^{th} discipline is given by the product of number of line searches and number of function evaluations to find a line search direction. Then the following relations for the number of function evaluations from [10] are given as below:

Let,

$$A = (n_g + n_{aux})$$

$$B_j = (n_{gj} + n_{auxj} + n_{Lj} + 1)$$

$$\text{Then for All-at-Once } n_f = (A + n_L) * B_j \quad (7.1)$$

Where,

n_f : total number of function evaluations

n_g : total number of input design variables which are shared by more than one discipline

n_{aux} : total number of coupling design variables

n_L : total number of local design variables

suffix 'j' stands for number of such variables required in the j^{th} discipline

For CO, the maximum number of line searches at the system level is equal to the number of system level design variables: $n_g + n_{aux}$. Assuming that the system level gradient information is obtained analytically using the post-optimality equation, then the number of function evaluations of the j^{th} discipline to find a stationary point at the system level is

$$n_f = A[B_j (B_j - 1)] \quad (7.2)$$

Then the ratio of the function evaluations for All-at-Once and CO for the j^{th} discipline is

$$F_{all-at-once} / F_{CO} = (A + n_L) / [A (B_j - 1)] \quad (7.3)$$

From (7.3), following observation can be inferred. Consider a single discipline problem. If it is decomposed in such that the number of local variables is large, then

$$(n_g + n_{aux} + n_L) \approx (n_{g1} + n_{aux1} + n_{L1}) \text{ and}$$

Then from 7.3

$$F_{single} / F_{bi} \approx 1 / A$$

This implies that in bi-level formulations for every change in disciplinary level, the number of function evaluations has to be performed 'A' times ie, for all the system level variables. Hence, the number of function evaluations in bi-level is 'A' times more compared to that of single level. However, it may be noted that bi-level formulations offer advantage of incorporating practical features like the distributed nature of organizational structure, in the formulation.

8. Conclusions

An overview of various types of Multi-disciplinary Design Optimization (MDO) Architectures has been presented in this report and attempt has been made to present a uniform notation for the various formulations. The desire to include system analysis even in the early stages of the design has prompted the development of MDO. The architecture or the formulation enables formal mathematical statement of the design problem. Several formulations have emerged over the last decade, each differing from the other in the manner in which the Multi-disciplinary Analysis (MDA) of the coupled system is executed and the manner in which the coupled system is decomposed. The basis for taxonomy of the formulations was highlighted. Among the single-level approaches, Individual Discipline Feasible formulation reflected an “in-between” strategy as compared to the extreme approaches adopted in Multi-Discipline Feasible and All-at-Once formulations. While the former strategy demanded a complete inter-disciplinary consistent solution at each optimization iteration, no such requirement was needed in the latter formulation. Design and optimization occur simultaneously in All-at-Once, thereby reducing the computational cost. Bi-level approaches, typically decomposed the design problem into a system level optimization problem and several sub-space optimization problems. It is believed that such a approach is closely analogous to the industry design environment. In Collaborative architecture, the problem was decomposed along disciplinary boundaries and each discipline was given the freedom to choose its own value of inter-disciplinary variable. Consistency was ensured at the system level, by imposing the discrepancy between the interdisciplinary variable value and the system target value as an equality constraint and requiring it drive to zero. In Concurrent Sub-Space Optimization formulation, no special distinction was made between the system level optimization and the sub-space optimization problems. In fact, the objective function for the system level and sub-space level was same. The constraints were also maintained the same. However, each sub-space used detailed analysis for states local to it, while for the non-local states an approximate model was used. Decomposition of the problem into a system level problem and several sub-discipline level problems enabled the disciplinary experts to contribute in the overall decision-making process. However, it was observed that decompositions could also lead to some problems as

reported in Collaborative Architecture. This problem was briefly discussed and it brought out the importance of formulation on the ability of the optimizer (gradient-based) to return proper solutions. Decomposition introduces certain aspects, which are not a part of the original problem statement, and hence utmost care has to be exercised so as not to cause any problems with regard to use of optimizer. Computational performance of a single level and a bi-level formulation was also discussed. It was observed that bi-level formulations with large number of disciplinary variables and a few system level variables would require more number of function evaluations than single level formulation. This is so, since every function evaluation in disciplinary level has to be repeated for all system design variables. However, CO allows disciplinary participation in the decision making process and provides disciplinary autonomy. Such features offer practical advantage with respect to the industry organizational structure. Absence of these features in single level formulations makes it difficult to implement them for industry type problems.

List of References

- [1.] N. M. Alexandrov and R.M. Lewis “*Comparative Properties of Collaborative Optimization and Other Approaches to MDO*” First ASMO UK/ISSMO Conference on Engineering Design Optimization, July,1999
- [2.] R. J. Balling and J. Sobieszczanski-Sobieski “*Optimization of Coupled Systems: A critical overview of approaches*” NASA-CR-185019 December,1994
- [3.] Robert Braun “*Collaborative Optimization : An Architecture for Large Scale Distributed Design*” Ph.D Dissertation, Stanford University, May,1996
- [4.] N. M. Alexandrov and R. M. Lewis “*Analytical and Computational Properties of Distributed Approaches to MDO*” AIAA2000-4718
- [5.] E. J. Cramer, Paul D. Frank and Gregory R. Shubin “*On Alternative Problem Formulations for Multidisciplinary Design Optimization*” AIAA 92-4752
- [6.] N. M. Alexandrov and R. M. Lewis “*Analytical and Computational properties of Collaborative Optimization*” NASA-TM-2000-210104
- [7.] R.S. Sellar, S.M. Batill and J.E. Renaud “*Response Surface Based, Concurrent Subspace Optimization for Multi-disciplinary System Design*” AIAA 96-0714
- [8.] J. Sobieszczanski-Sobieski “*Bi-level Integrated System Synthesis*” NASA Langley Research Center. AIAA Journal Vol38 No1, Jan2000
- [9.] R.J. Balling and C.A. Wilkinson “*Execution of Multidisciplinary Design Optimization Approaches on Common Test Problems*” Brigham Young University, Utah. AIAA Journal Vol.35 No1, Jan1997
- [10.] Ian P. Sobieski “*Multidisciplinary Design Using Collaborative Optimization*” Ph.D. Dissertation , Stanford University, August 1998